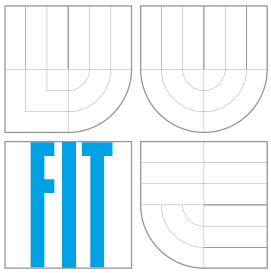
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

BIOINFORMATICKÝ NÁSTROJ PRO PREDIKCI ROZPUSTNOSTI PROTEINŮ

BIOINFORMATICS TOOL FOR PREDICTION OF PROTEIN SOLUBILITY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PATRIK HRONSKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. TOMÁŠ MARTÍNEK, Ph.D.

BRNO 2016

Zadání diplomové práce/18179/2015/xhrons00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Hronský Patrik, Bc.**
Obor: Bioinformatika a biocomputing
Téma: **Bioinformatický nástroj pro predikci rozpustnosti proteinů**
Bioinformatics Tool for Prediction of Protein Solubility
Kategorie: Bioinformatika

Pokyny:

1. Seznamte se s problematikou rozpustnosti proteinů a existujícími algoritmy pro její predikci.
2. Vytvořte trénovací a testovací datovou sadu rozpustných a nerozpustných proteinů na základě dostupných databází, patentů či literárních rešerší.
3. Navrhněte vhodný nástroj pro predikci rozpustnosti proteinů a proveďte jeho implementaci.
4. Činnost nástroje zhodnoťte na testovací datové sadě, kterou použijte rovněž pro ohodnocení existujících nástrojů.
5. Zhodnoťte dosažené výsledky a diskutujte možnosti dalšího pokračování projektu.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Martínek Tomáš, Ing., Ph.D.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Táto diplomová práca sa zaoberá rozpustnosťou rekombinantných proteínov a hlavne jej predikciou. Približuje čitateľovi problematiku vzniku proteínov, ako aj proces tvorby rekombinantných proteínov. Syntetická výroba rekombinantných proteínov je významným prínosom napríklad pre farmakologický priemysel. Ide však o náročný proces, ktorý navyše nemusí priniesť použiteľné proteíny. Podstatným ukazateľom použiteľnosti výsledných proteínov je práve ich rozpustnosť. Pre spoločnosti, zaoberajúce sa výrobou rekombinantných proteínov, je samozrejme výhodné sústrediť svoje úsilie a prostriedky na výrobu tých proteínov, ktoré budú v praxi použiteľné. V tomto ohľade podáva pomocnú ruku bioinformatika, ktorá je pomocou techník strojového učenia schopná predikovať rozpustnosť proteínov, napríklad na základe ich sekvencie. Táto práca zoznamuje čiteteľa so základnými princípmi strojového učenia a predstavuje niektoré metódy strojového učenia, používané okrem iného aj na predikciu rozpustnosti proteínov. Práca ďalej opisuje vytvorenú dátovú sadu, ktorá je neskôr použitá na testovanie vybraných prediktorov a pre trénovanie konsenzuálneho prediktoru, ktorý je cieľom práce. Ďalej sa práca zameriava na konkrétne existujúce prediktory rozpustnosti proteínov a prezentuje základné princípy ich funkčnosti, ako aj výsledky ich testovania. V závere prezentuje konsenzuálny metaprediktor rozpustnosti proteínov.

Abstract

This master's thesis addresses the solubility of recombinant proteins and its prediction. It describes the subject of protein synthesis, as well as the process of recombinant protein creation. Recombinant protein synthesis is of great importance for example to pharmacologic industry. This synthesis is not a simple task and it does not always produce viable proteins. Protein solubility is an important factor, determining the viability of the resulting proteins. It is of course favourable for companies, that take part in recombinant protein synthesis, to focus their effort and their resources on proteins, that will be viable in the end. In this regard, bioinformatics is of great help, as it is capable, with the help of machine learning, of predicting the solubility of proteins, for example based on their sequences. This thesis introduces the reader to the basic principles of machine learning and presents several machine learning methods, used in the field of protein solubility prediction. It deals with the definition of a dataset, which is later used to test selected predictors, as well as to train the ensemble predictor, which is the main focus of this thesis. It also focuses on several specific protein solubility predictors and explains the basic principles upon which they are built, as well as the results of their testing. In the end, it presents the ensemble predictor of protein solubility.

Kľúčové slová

proteín, strojové učenie, rozpustnosť, predikcia, klasifikácia, rekombinantný, konsenzuálna metóda

Keywords

protein, machine learning, solubility, prediction, classification, recombinant, ensemble method

Citácia

HRONSKÝ, Patrik. *Bioinformatický nástroj pro predikci rozpustnosti proteinů*. Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Martínek Tomáš.

Bioinformatický nástroj pro predikci rozpustnosti proteinů

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Tomáša Martínka, Ph.D., a že som uviedol všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Patrik Hronský
22. mája 2016

Poďakovanie

Ďakujem pánovi Ing. Tomášovi Martínkovi, Ph.D., za odbornú pomoc pri riešení diplomovej práce. Ďakujem taktiež Národnej Gridovej Infraštruktúre MetaCentrum, poskytovanej v rámci programu „Projekty velkých infrastruktur pro VaVaI“ (LM2010005), za prístup k výpočtovým a pamäťovým zdrojom.

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

Jedným z mnohých prínosov, ktoré biotechnológia prináša do rôznych oblastí života, je tvorba rekombinantných proteínov. Ide o syntetickú výrobu proteínov, ktoré následne nachádzajú svoje uplatnenie napríklad vo farmakologickom priemysle, ale aj v iných oblastiach priemyslu, kde sú využívané biochemické procesy. Táto syntetizácia prebieha pomocou špecializovaných baktérií, ako napríklad *Escherichia coli*, do ktorých je prostredníctvom klonovacieho vektoru zavedená rekombinantná DNA, z ktorej má následne vzniknúť požadovaný proteín. Očividne teda ide o neľahký proces a aj napriek pokrokom v genetickom inžinierstve je tvorba rekombinantných proteínov časovo a finančne náročná. K tomu prispieva okrem iného aj fakt, že výsledný proteín nemusí byť vôbec v praxi použiteľný, pretože môže dochádzať k nesprávnemu zloženiu proteínov do požadovanej štruktúry. A práve *rozpustnosť proteínu* je faktor, ktorý veľkou mierou rozhoduje o schopnosti proteínu zložiť sa do správnej 3D štruktúry a v konečnom dôsledku tak vypovedá o kvalite a použiteľnosti daného proteínu.

Mieru rozpustnosti proteínu je samozrejme možné zistiť pokusným vytvorením daného proteínu, tento proces je však, ako sme spomenuli vyššie, časovo a finančne náročný, a skúmať takto metódou pokus-omyl je jednoducho nevýhodné. Tu podáva pomocnú ruku bioinformatika a strojové učenie, ktoré sa snaží predikovať rozpustnosť výsledného proteínu na základe poskytnutej sekvencie aminokyselín daného proteínu. Spoločnosti zaoberajúce sa syntetizáciou proteínov tak môžu venovať svoje úsilie a prostriedky proteínom, pri ktorých je väčšia šanca, že budú v konečnom dôsledku použiteľné.

Za posledné roky bolo vytvorených viacero takýchto prediktorov, ktoré skúmajú aminokyselinovú sekvenciu proteínov a extrahujú z nej vlastnosti, na základe ktorých následne predpovedajú rozpustnosť. Cieľom tejto práce je existujúce nástroje využiť a vybudovať nad nimi metaprediktor, ktorý taktiež využije metódy strojového učenia, ale ako svoj vstup využije výstupy existujúcich nástrojov, aby z nich vytvoril konsenzus. Myšlienkou, ktorá za touto prácou stojí, je, že rôzne prediktory môžu predikovať rozpustnosť s rôznou spoľahlivosťou pre rôzne typy proteínov, a práve metaprediktor by sa mal naučiť rozoznávať, ktorý prediktor predikuje správne pre ktoré proteíny, a na základe tejto vedomosti vytvoriť jeden výstupný verdikt z viacerých poskytnutých a tak v konečnom dôsledku zvýšiť presnosť predikcie.

V kapitole 2 sa zaoberáme proteínmi a procesom tvorby rekombinantných proteínov. Kapitola 3 poskytuje informácie o strojovom učení a opisuje techniky použité v prediktoroch rozpustnosti. Ďalšia kapitola sa potom zaoberá použitými dátovými sadami, ktoré sme využili na testovanie existujúcich prediktorov, ako aj na učenie a testovanie výsledného metaprediktoru. Následne, v kapitole 5, sa zaoberáme konkrétnymi použitými prediktormi

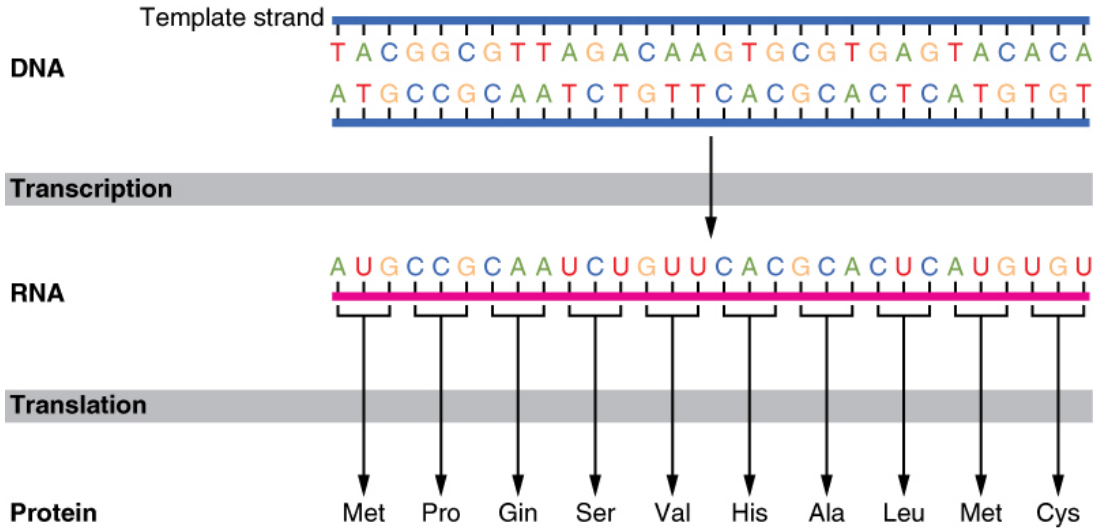
a ich vlastnosťami. Nasledujúca kapitola 6 pojednáva o celkovom procese práce a poskytuje implementačné detaily k niektorým krokom. Táto kapitola taktiež opisuje tvorbu nášho metaprediktoru, ako aj jeho testovanie. V závere zhodnocujeme dosiahnuté výsledky a poskytujeme odôvodnenie týchto výsledkov.

Kapitola 2

Proteíny

Proteíny sú makromolekuly, zložené z množstva aminokyselín, plniace v organizme rôzne úlohy: slúžia ako stavebný materiál, senzory prenášajúce signály, presúvajú molekuly po organizme alebo katalyzujú metabolické reakcie [34].

Proces, pri ktorom sa aminokyseliny viažu na seba, aby vytvorili proteíny, sa nazýva *translácia*. Aminokyseliny sú do vznikajúceho reťazca vyberané na základe trojíc nukleotidov mRNA, kde každá takáto trojica (zvaná *kodón*) kóduje konkrétnu aminokyselinu. RNA sa skladá zo štyroch rôznych typov nukleotidov (cytozín, guanín, adenín a uracil) a existuje teda $4^3 = 64$ možných trojíc, ktoré je možné z nich vytvoriť. Týchto 64 kodónov však kóduje spolu 20 rôznych aminokyselín[1], takže niektoré aminokyseliny sú kódované viacerými rôznymi kodónmi. Proces transkripcie a translácie je zobrazený na obrázku 2.1[2].



Obr. 2.1: Transkripcia a translácia.

Keďže sa aminokyseliny viažu na základe kodónov z mRNA a mRNA vzniká z DNA v procese zvanom *transkripcia*, závisí tvorba proteínov priamo na postupnosti nukleotidov v DNA. Teoreticky je teda možné zavedením umelo vytvorenej DNA do organizmu vynútiť

[1] niektoré zdroje uvádzajú 22, pridávajúc selenocysteín a pyrolyzín

[2] http://oerpub.github.io/epubjs-demo-book/content/m46032.xhtml

vznik nami požadovaného proteínu. A práve na tomto princípe funguje tvorba rekombinantných proteínov.

2.1 Tvorba rekombinantných proteínov

Termín *rekombinantný proteín* sa používa pre označenie proteínov, ktoré vznikli expresiou génov *rekombinantnej DNA*. Rekombinantná DNA (niekedy nazývaná aj *chimerická DNA*) je typ DNA, ktorý vznikol kombináciou aspoň dvoch rôznych vlákien DNA. Tieto vlákna môžu pochádzať z ľubovoľných organizmov (napr. je možné spájanie bakteriálnej DNA s DNA človeka) a prípadne môže ísť aj o umelo vytvorené vlákna DNA [34].

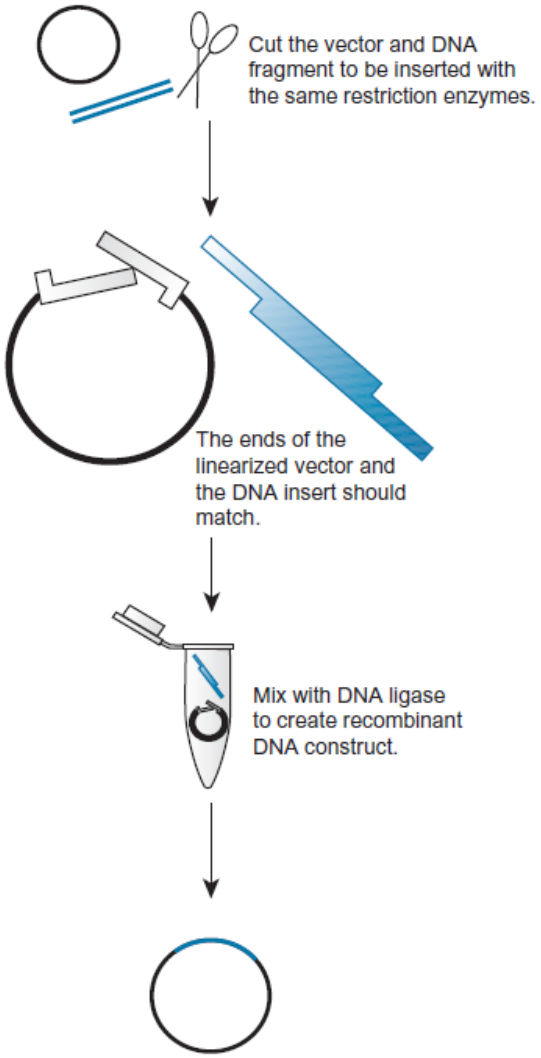
Po získaní požadovaných vlákien DNA je nutné vytvoriť rekombinantnú DNA, kde dané vlákna spojíme. Na to slúžia *DNA vektory*. Vektor je molekula DNA, ktorá je schopná niesť izolovanú sekvenciu DNA a dokáže sa vo vhodnom hostiteľskom organizme autonómne replikovať. Hlavnými typmi vektorov sú *plazmidy*, *vírusové vektory*, *cosmidy* a *umelé chromozómy*, najčastejšie sa využívajú plazmidy [34]. Všetky typy vektorov majú niekoľko spoločných rysov:

- *počiatok replikácie* je špecifická sekvencia DNA, ktorá slúži ako miesto, kde sa začína replikácia DNA. Bez tejto sekvencie by nedochádzalo k množeniu vektoru,
- *restrikčné miesta* slúžia na vkladanie DNA do vektoru. Vektory často obsahujú oblasť zvanú *multiple cloning site*, ktorá obsahuje väčší počet restrikčných miest, čím umožňuje relatívne jednoduché vkladanie vlákna DNA do vektoru,
- *selekčné markery* sa používajú na výber pozitívne transformovaných buniek, teda na výber tých buniek, v ktorých sa úspešne uchytila umelo vložená DNA. Ako takýto marker sa často používa odolnosť k antibiotikám: vzniknuté bunky sa umiestnia do prostredia s obsahom antibiotických látok a v takomto prostredí budú schopné prežiť len tie bunky, ktoré majú spomínanú odolnosť k antibiotikám. Tým vylúčime bunky, v ktorých nedošlo k pozitívnej transformácii po vložení vektoru.

Pri vkladaní DNA do vektoru sa ako vektor, tak aj vkladaný DNA fragment naštiepia pomocou rovnakých restrikčných enzýmov tak, aby si prevísajúce časti DNA a vektoru odpovedali. Pomocou DNA ligázy sú prevísajúce konce spojené a je opäť vytvorený celistvý vektor, ktorý však už obsahuje vloženú DNA [5]. Tento proces je ukázaný na obrázku 2.2.

Častým spôsobom tvorby rekombinantných proteínov je teda vytvorenie vektorov s rekombinantnou DNA napríklad vo forme plazmidu a následné vloženie plazmidu do bunky hostiteľského organizmu, často *E. coli*. Takto upravené bunky sa následne množia a tým sa množí aj umelo vložený gén. Cieľom však nie je naklonovať veľké množstvá génu, ale získať výsledné proteíny, ktoré z daného génu majú vzniknúť. Za týmto účelom sa do vektoru okrem daného génu vkladajú aj sekvencie, ktoré napomáhajú expresii génov (ako napríklad promótory), a takto upravené vektory sa nazývajú *expresné vektory* [25].

Pri tvorbe rekombinantných proteínov sa využívajú rôzne hostiteľské organizmy, každý so svojimi výhodami a nevýhodami. Z prokaryotických organizmov je veľmi bežné použitie baktérie E. coli. Jej molekulárna biológia, biochémia a fyziológia sú veľmi dobre zdokumentované a baktériu je možné jednoducho množiť. Medzi problémy patrí neschopnosť E. coli efektívne tvoriť veľmi dlhé alebo krátke proteíny. Nedochádza k posttranslačným úpravám, ako glykozylácii alebo acylácii. Navyše vytvárajú baktérie pyrogény a endotoxíny, ktoré je nutné z proteínov úplne odstrániť pred vložením do zvierat alebo ľudí. Ďalšie problémy



Obr. 2.2: Tvorba rekombinantnej DNA [5].

s E. coli spočívajú v rôznorodosti úrovne expresie, rozpustnosti a proteínovej purifikácie, existujú však rôzne metódy genetickej manipulácie, ktoré sa snažia tieto problémy riešiť (napríklad používanie silných promótorov) [27].

Ďalšou možnosťou je využitie kvasiniek. Prvou kvasinkou, upravovanou pre účely expresie rekombinantných proteínov, bola *Saccharomyces cerevisiae* (kvasinka pivná). Vďaka jej výdatnému používaniu sme získali bohaté vedomosti o molekulárnej biológii a fyziológii tejto kvasinky. Podobne ako iné eukaryoty, kvasinky sú schopné väčšiny posttranslačných úprav, ktoré sú typické pre bunky cicavcov. Vzhľadom k mimobunkovým proteázam a odlišnostiam v glykozylácii pri proteínoch, exprimovaných kvasinkami, je však použitie takýchto proteínov problematické, pretože napríklad môžu pri cicavcoch spôsobovať imunitnú reakciu. Pokiaľ je však nutné produkovať proteíny, ktoré nevyžadujú glykozyláciu, aká prebieha pri cicavcoch, a ktoré sú odolné voči proteázam, kvasinky sú vhodným hostiteľom. Významným príkladom takéhoto proteínu je inzulín [27].

Na expresiu rekombinantných proteínov sa využívajú aj živočíšne bunky. Genetická manipulácia takýchto buniek je samozrejme zložitejšia, než napríklad manipulácia baktérií. Využívanie živočíšnych buniek je však na vzostupe, pretože mnoho proteínov s farmaceutickým využitím vyžaduje komplexnú glykozyláciu, ktorú nie je možné prakticky vykonávať prostredníctvom prokaryotických hostiteľov, prípadne pomocou nižších eukaryotov. Problémom práce s živočíšnymi bunkami sú okrem iného komplexné požiadavky na výživu týchto buniek. Na dodanie potrebných hormónov a rastových faktorov sa využívalo hovädzie sérum, to však spôsobovalo problémy s kontamináciou výsledného produktu vírusmi a s extrakciou proteínov z média, obsahujúceho sérum. V dôsledku toho vznikli médiá, ktoré nevyužívajú séra. Ďalším problémom je prenos génov do buniek. V tejto oblasti sa využívajú napríklad vírusové vektory [27].

Počas života hostiteľa teda dochádza k transkripcii a translácii vloženého génu a vznikajú tak požadované proteíny. Ako sme už spomenuli v úvode, proteíny sa ešte musia zložiť do správnej štruktúry, aby mohli správne plniť svoju funkciu. V niektorých prípadoch vytvárajú proteíny *bunkové inklúzie*, teda zhluky nesprávne zložených proteínov. Toto často nastáva práve pri expresii umelo vložených génov, ktoré by inak v danom hostiteľskom organizme nevznikli, prípadne nevznikali v tak veľkých množstvách. S týmito zhlukmi proteínov je síce možné pracovať a pomocou zvýšenia rozpustnosti a následnej renaturácie je možné z nich vytvoriť funkčné proteíny, tieto operácie sú však náročné a často nevedú k použiteľným výsledkom. Práve preto sú uprednostňované proteíny, ktoré sú po vzniku rozpustné a teda netvoria inklúzie [27]. Ako sme spomenuli už v úvode, na predikciu toho, ktoré proteíny budú rozpustné, sa využíva strojové učenie.

Kapitola 3

Metódy strojového učenia

Cieľom strojového učenia je vo všeobecnosti získať užitočné informácie z poskytnutých dát na základe dobre vybudovaného pravdepodobnostného modelu. Hlavnou myšlienkou je automatizovať proces tvorby tohto modelu a charakterizovať ho množstvom parametrov, z ktorých počítač zostaví tie najvhodnejšie. Táto oblasť informatiky nachádza veľkú inšpiráciu vo svojom biologickom vzore – mozgu. Rôzne prístupy strojového učenia nachádzajú svoje uplatnenie najmä v odvetviach, ktoré poskytujú veľké množstvo dát, o ktorých ale nemáme príliš bohaté vedomosti [4]. Pojem *učenie* môžeme v kontexte strojového učenia chápať vo význame *budovanie modelu*, teda zisťovanie, ako nastaviť parametre modelu tak, aby výsledky, dosiahnuté počítačom, zodpovedali očakávaným výsledkom.

3.1 Typy strojového učenia

Úlohou strojového učenia je teda zistiť spôsob, ako z poskytnutých vstupných dát získať dáta výstupné, bez toho, aby mu vývojári explicitne stanovili algoritmus tejto transformácie. Počas učenia musí počítač dostávať nejakú spätnú vazbu, na základe ktorej dokáže zhodnotiť, či sa správne naučil riešiť zadanú úlohu. Na základe typu tejto spätnej väzby rozišujeme tri typy strojového učenia [28]:

1. *učenie s učiteľom* predstavuje situácie, keď sú k dostupným vstupným dátam priradené aj správne (očakávané) výstupy a počítač musí nájsť správnu funkciu, ktorou bude vstupné dáta transformovať na správne výstupné.

2. *učenie bez učiteľa* je prípad, keď je nutné nájsť vo vstupných dátach určité vzory, avšak nie sú poskytnuté žiadne informácie o tom, čo je považované za korektný výstup.

3. *posilované učenie* je najvšeobecnejší typ strojového učenia, kde počítač nedostáva informáciu o tom, ako dobre vykonáva úlohu od učiteľa, ale musí si toto hodnotenie odvodiť sám na základe prostredia.

Všetky metódy v tejto diplomovej práci sú predstaviteľmi učenia s učiteľom, preto sa zameriame na tento typ.

3.2 Učenie s učiteľom

Pri učení s učiteľom dostáva počítač na vstupe *trénovaciu sadu*, ktorá pozostáva zo vstupných vzorkov, kde každý má priradenú aj korektnú výstupnú hodnotu. Počítač potom ur-

čitým algoritmom, ktorý sa líši pre každú metódu strojového učenia, aktualizuje tvorený model a upravuje jeho parametre tak, aby tento model čo najpresnejšie (s čo najväčšou úspešnosťou) transformoval vstupné dáta na očakávané výstupné hodnoty.

Bežným problémom strojového učenia je tzv. *preučenie*, čo je situácia, keď počítač generuje vynikajúce výsledky pre trénovaciu sadu, avšak pre nové vstupy (také, ktoré počas učenia počítač nemal k dispozícii) sa presnosť naučeného modelu značne znižuje. Dôvodom tohto problému je, že počítač sa príliš sústredil na rysy trénovacej sady a objavil súvislosti, ktoré v danej dátovej sade síce náhodou korelujú so správnym výsledkom, avšak v danom type dát všeobecne nesúvisia so správnym výstupom, a preto bude takýto model zlyhávať na dátach z reálneho sveta.

Opačným prípadom je, keď je počítač schopný *dobre generalizovať*, teda reaguje správne aj na nové hodnoty [29]. Takýto model odhalil tie správne súvislosti medzi rysmi namiesto tých, ktoré platia len v trénovacej sade.

Na základe typu výstupu môžeme uvažovať ďalšie delenie strojového učenia s učiteľom, a to na:

1. *klasifikáciu*, ktorá ako výstup generuje jednu z konečného množstva hodnôt. Pomerne častá je binárna klasifikácia, ktorá nad vstupnými dátami vytvorí verdikt nadobúdajúci len dve hodnoty – kladnú a zápornú. Metódy, ktorými sa budeme zaoberať v tejto práci, sú práve tohto typu (binárne klasifikujú proteíny ako rozpustné alebo nerozpustné),

2. *regresiu*, ktorá generuje číslo z neobmedzeného intervalu.

3.3 Metódy klasifikácie

Keďže cieľom tejto práce je binárne klasifikovať proteíny ako rozpustné alebo nerozpustné, zameriame sa na klasifikačnú časť metód strojového učenia. Klasickým príkladom klasifikácie je príklad s bankou a rizikom pôžičky. Ide o príklad, kde klient príde do banky so žiadosťou o pôžičku. Úlohou strojového učenia je tu na základe informácií o klientovi usúdiť, či bude klient poctivo a včas splácať pôžičku. Počítač teda na vstupe dostane jednak informácie o klientovi, ako výšku príjmu, druh zamestnania, vek, finančnú históriu a podobne, a zároveň má prístup k historickým dátam o predchádzajúcich prípadoch, keď si klienti vyberali pôžičky, pričom pri týchto prípadoch pozná jednak vlastnosti týchto klientov, ale aj informáciu o tom, ako dobre pôžičku splatili. Na základe týchto historických dát musí počítač odvodiť model tak, aby pre určité vlastnosti klienta získal na výstupe určitý verdikt o jeho schopnosti splácať pôžičku, ktorý reálne zodpovedá historickým skúsenostiam. Následne už musí len aplikovať tento model na novú žiadosť a na základe vstupných informácií predpovedať, či bude klient schopný pôžičku splácať, alebo nie [3].

V tomto príklade je teda *trénovacou sadou* množina historických dát, pričom každý historický prípad pozostáva z dvoch častí – vstupov a výstupnej triedy. Vstupom sú v tomto prípade informácie o klientovi, ako boli spomenuté vyššie. Výstupnou triedou je schopnosť klienta splácať úver a môže ísť napr. o jednu z dvoch hodnôt. Na tejto trénovacej sade sa vytvorí model. Na základe tvorby modelu rozlišujeme rôzne metódy klasifikácie.

3.3.1 Bayesovské klasifikátory

Bayesovské klasifikátory predstavujú časť strojového učenia postavenú na pravdepodobnostnom modelovaní. V mnohých prípadoch nepoznáme všetky okolnosti, ktoré ovplyvňujú

jav, ktorý študujeme. V takýchto situáciách nemôžeme s určitosťou tvrdiť, že daný jav nastane alebo nenastane, ale môžeme využiť pravdepodobnosť a predpovedať, s akou pravdepodobnosťou daný jav nastane alebo nenastane.

V tejto oblasti sa pracuje s pojmom *podmienená pravdepodobnosť*, kde zápis $P(X|Y)$ znamená pravdepodobnosť javu X, pokiaľ vieme, že nastal jav Y. Platí tu vzťah $P(X \wedge Y) = P(X|Y)P(Y)$, teda pravdepodobnosť, že nastanú súčasne javy X aj Y sa rovná násobku pravdepodobnosti, že nastane jav X pokiaľ nastal jav Y, a pravdepodobnosti javu Y. Tento vzťah samozrejme platí aj vo forme $P(X \wedge Y) = P(Y|X)P(X)$ a na základe týchto dvoch vzorcov vznikol Bayesov vzorec [29]:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \tag{3.1}$$

Ako príklad použitia si predstavme snahu diagnostikovať pacienta. Jav X bude predstavovať to, že pacient trpí meningitídou, jav Y bude znamenať prítomnosť symptómu stuhnutej šije. Cieľom diagnózy je zistiť, aká je pravdepodobnosť, že pacient trpí meningitídou, pokiaľ vieme, že má stuhnutú šiju. Podľa vzorca to zistíme podelením pravdepodobnosti, že sa meningitída a stuhnutá šija vyskytujú súčasne, a nepodmienenej pravdepodobnosti, že ľubovoľný človek má stuhnutú šiju.

Povedzme, že sa stuhnutá šija objavuje pri meningitíde s pravdepodobnosťou 50%, nepodmienená pravdepodobnosť výskytu meningitídy je $\frac{1}{50000}$ a nepodmienená pravdepodobnosť stuhnutej šije je $\frac{1}{20}$. Po dosadení do vzorca teda získavame pravdepodobnosť [28]:

$$P(\text{meningitída}|\text{stuhnutá šija}) = \frac{0.5 * \frac{1}{50000}}{\frac{1}{20}} = 0.0002 \tag{3.2}$$

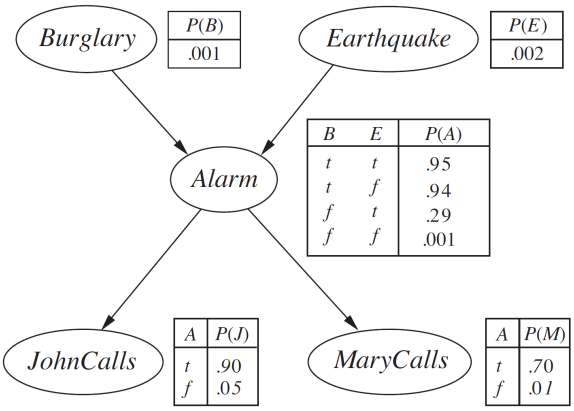
Pri klasifikácii máme na vstupe určitú vzorku dát Z a cieľom je zaradiť ju do jednej zo skupín C_1 až C_n. Danú vzorku zaradíme do triedy C_i, pokiaľ platí, že $P(C_i|Z)$ je maximálna, teda pravdepodobnosť, že ak má daný vzorok vlasnosť Z, tak bude patriť do skupiny C_i je väčšia, než všetky ostatné pravdepodobnosti pre ostatné triedy C. Vzhľadom na to, že menovateľ vyššie uvedeného Bayesovho vzorca $P(Y)$ je konštanta, hľadáme maximum čitateľa, teda maximálnu hodnotu pravdepodobnosti, že zároveň nastal jav X a jav Y. Konkrétne teda hľadáme maximum pravdepodobnosti, že vzorka dát Z patrí do triedy C_i cez všetky i [23].

Práve opísaná metóda sa nazýva *naivný bayesovský klasifikátor*, kde slovo *naivný* je použité preto, pretože tento typ klasifikácie predpokladá nezávislosť rôznych rysov, z ktorých je vzorok Z utvorený. Takúto nezávislosť však v praxi príliš často nenachádzame. Riešením tohto problému sú *bayesovské siete*, ktoré predstavujú acyklické grafy, v ktorých náhodné premenné predstavujú uzly. Hrany medzi uzlami predstavujú závislosti premenných, a ak vedie hrana z uzlu X do uzlu Y, hovoríme, že uzol X je rodičom uzlu Y a uzol Y je označený pravdepodobnosťou $P(Y|X)$, ktorá označuje vplyv rodiča na uzol. Príklad takejto siete je na obrázku 3.1 [29].

Uzly *Burglary* a *Earthquake* sú nezávislé, zatiaľ čo ostatné uzly sú závislé na svojich predchodcoch. Tento model je v reálnom svete praktickejší, avšak jeho zostrojenie vyžaduje znalosti závislostí medzi premennými a teda vyžaduje určitý zásah experta.

3.3.2 Rozhodovacie stromy

Rozhodovací strom predstavuje funkciu, ktorá na vstupe dostane vektor hodnôt a produkuje na výstupe rozhodnutie, teda jedinú hodnotu. Táto hodnota, ako aj hodnoty vstupov,



Obr. 3.1: V bayesovskej sieti sú brané v úvahu závislosti medzi premennými. Na obrázku napríklad udalosť *Burglary* nie je závislá na žiadnej inej a má priradenú nepodmienenú pravdepodobnosť 0.001. Oproti tomu udalosť *Alarm* závisí od udalostí *Burglary* a *Earthquake*, a príslušná tabuľka zobrazuje podmienené pravdepodobnosti, kde napríklad prvý riadok ukazuje pravdepodobnosť, že dôjde k udalosti *Alarm*, pokiaľ došlo k udalosti *Burglary* a zároveň k udalosti *Earthquake*, pričom táto pravdepodobnosť je 0.95. Druhý riadok zas zobrazuje pravdepodobnosť javu, pokiaľ došlo k udalosti *Burglary*, avšak nedošlo k udalosti *Earthquake*. Podobne aj udalosť *JohnCalls* je závislá, tentokrát na udalosti *Alarm*, a obdobne obsahuje podmienené pravdepodobnosti v závislosti na udalosti *Alarm*.

môže byť diskrétna alebo spojitá. Keďže v prípade predikcie rozpustnosti ide o binárnu klasifikáciu, zameriame sa na túto oblasť.

Aby sa dostal k výsledku, musí rozhodovací strom prechádzať svojimi uzlami od koreňa k listom. Každý uzol predstavuje otestovanie jedného zo vstupných atribútov a na základe výsledku tohto testu sa presunie po určitej hrane do určitého synovského uzlu. Každá hrana z rodičovského uzlu do synovského uzlu predstavuje jednu z možných hodnôt skúmaného vstupného atribútu. Pokiaľ nadobúda vstupný atribút neobmedzený počet možných hodnôt (reálne číslo, neobmedzené celé číslo, atď.) alebo je počet možných hodnôt príliš vysoký, jednotlivé hrany nepredstavujú presné hodnoty, ale určité rozsahy hodnôt, teda dochádza k diskretizácii spojitej premennej. V každom listovom uzle sa potom nachádza výsledné rozhodnutie.

Zostrojenie takého modelu však nie je triviálne a je nutné použiť rozumné heuristiky a algoritmy, pretože množina všetkých potenciálnych stromov rastie s pribúdajúcimi vstupnými atribútmi geometrickou radou. V praxi sa snažíme vytvárať stromy čo najmenšie, čo dosiahneme tak, že testovanie najvýznamnejších vstupných atribútov umiestnime čo najbližšie ku koreňu stromu. Za najvýznamnejší atribút považujeme ten, ktorý najjasnejšie rozdeľuje vstupy do výstupných kategórií [29].

Pri tvorbe stromu sa na prevenciu preučenia využíva metóda *prerezávania* (angl. *pruning*), a to buď prerezávania počas tvorby stromu (*prepruning*) alebo prerezávania už vytvoreného stromu (*postpruning*). Často sa tieto dva prístupy kombinujú. *Prepruning* spočíva v zabránení generovania vetiev, ktoré by už obsahovali príliš malú časť trénovacích dát. *Postpruning* naopak pracuje už s vytvoreným stromom, a hľadá podstromy, ktoré vedú k zlému zovšeobecňovaniu modelu. Pracuje tak, že na začiatku učenia vyberie podmnožinu trénovacích dát, ktorá sa pri trénovaní nepoužije. Následne skúša nahradiť rôzne podstromy

listovým uzlom, obsahujúcim všetky trénovacie dáta, ktoré daný podstrom predstavoval. Potom testuje tento uzol na dátch, ktoré si na začiatku vyčlenil a pokiaľ dosahuje horšie výsledky než pôvodný podstrom, tento podstrom ponechá, inak ho vo výslednom strome nahradí listovým uzlom a tým strom zmenšuje [3].

Určitou nadstavbou nad rozhodovacími stromami sú takzvané *lesy rozhodovacích stromov* (*decision tree forests*). Pri tomto prístupe sa z množiny rysov vyberú rôzne podmnožiny a pre každú takúto podmnožinu sa vytvorí nový rozhodovací strom vyššie uvedenými postupmi. Vzhľadom na to, že je každý strom vybudovaný nad inými rysmi trénovacích dát, zovšeobecňuje každý strom inak. Metóda potom na základe výslednej klasifikácie jednotlivých stromov vytvorí finálnu klasifikáciu [16].

3.3.3 Lineárna regresia a klasifikácia

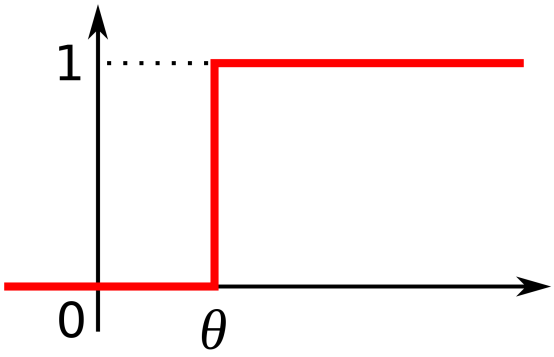
Podstatou lineárnej regresie je umiestniť do priestoru priamku tak, aby čo najlepšie aproximovala všetky body trénovacej množiny. Táto priamka je v priestore definovaná váhovým vektorom a práve výpočet tých správnych váh je cieľom algoritmov lineárnej regresie. Za najlepšie váhy sú považované tie, ktoré definujú priamku, dosahujúcu najmenšiu chybovosť. Chyba je často počítaná ako súčet druhých mocnín chýb pre jednotlivé trénovacie vektory, kde chyba predstavuje rozdiel skutočnej hodnoty a hodnoty získanej pomocou regresie [29].

Využitie regresie je, že po vytvorení modelu (teda nastavení správnych váh) môžeme za premenné dosadiť nové (pri trénovaní neznáme) hodnoty a na základe nich získať výstupnú hodnotu, teda reálne číslo. Podobný princíp sa však dá použiť aj na klasifikáciu a v tom prípade hovoríme o *logistickej regresii*.

Princíp je podobný ako pri klasickej lineárnej regresii, opäť je snaha naučiť sa na trénovacích dátach správne váhy. V tomto prípade však výstupom nie je nutne priamka v priestore, ale všeobecne hyperplocha, ktorej dimenzia závisí od veľkosti vstupných vektorov. Pre dvojrozmerné vstupné premenné by opäť išlo o priamku, avšak napríklad pri vstupoch s tromi rysmi by už išlo o plochu v trojrozmernom priestore. Táto hyperplocha sa nazýva *deliaca hyperplocha* a jej úlohou je rozdeliť vstupné dáta na dve skupiny podľa ich priradených tried.

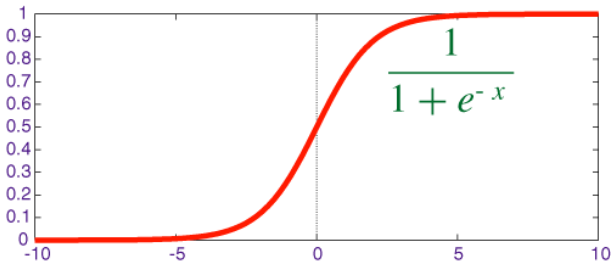
Ide o učenie s učiteľom, teda vstupné dáta musia mať okrem rysov stanovenú aj triedu. Chyba získanej hyperplochy sa potom prirodzene počíta ako počet trénovacích vzoriek umiestnených do zlej triedy. *Hraničná funkcia* je funkcia, ktorá ako svoj vstup prijíma násobok váh deliacej hyperplochy a vstupného vektoru a na výstupe poskytuje výslednú hodnotu. V prípade skokovej hraničnej funkcie je výstupom 0 alebo 1 čo sa dá predstaviť tak, že pokiaľ sa vstupný prvok ocitne na jednej strane deliacej hyperplochy, zaradíme ho do jednej triedy, pokiaľ na druhej strane, zaradíme ho do druhej triedy. Tento prístup jednak nedefinuje triedu pre prvky, ktoré ležia priamo na deliacej hyperploche a jednak nerozlišuje, či sa daný prvok nachádza v tesnej blízkosti deliacej hyperplochy, alebo vo väčšej vzdialenosti od nej. Taktiež o každom prvku rozhoduje s absolútnou istotou, teda tvrdí, že daný prvok patrí do danej triedy so 100% pravdepodobnosťou. Skoková hraničná funkcia je na obrázku 3.2.

Ako hraničná funkcia sa namiesto skokovej funkcie často používa *sigmoidálna funkcia* (na obrázku 3.3), vďaka čomu je výstupom klasifikácie číslo v rozsahu $< 0, 1 >$ namiesto dvoch hraničných čísel 0 a 1. Toto nám umožňuje chápať výstup klasifikácie ako pravdepodobnosť, že daný prvok patrí do triedy 1, čím získavame viac informácie, než pri tvrdej klasifikácii. Táto hraničná funkcia sa označuje ako funkcia s jemným prahom (angl. *soft threshold*), na rozdiel od predchádzajúceho prípadu, kde hovoríme o tvrdom prahu (angl. *hard*



Obr. 3.2: Skoková funkcia.

threshold) [29]. Použitím takejto funkcie navyše definujeme pravdepodobnosť príslušnosti k triede aj pre prvky, ležiace priamo na deliacej hyperploche, a to 50% (čo teda znamená, že je rovnako pravdepodobné, že prvok patrí do jednej aj do druhej triedy).



Obr. 3.3: Sigmoidálna funkcia.

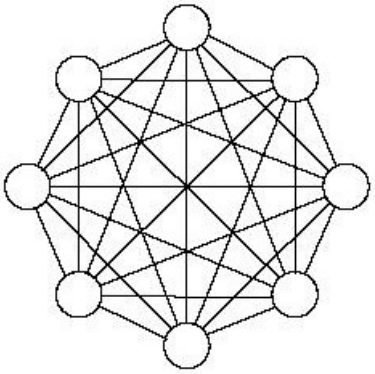
3.3.4 Neurónové siete

Neurónové siete vznikli pôvodne s cieľom modelovať spracovávanie informácií a učenie, ktoré prebiehajú v mozgu. Mozog je zložený z približne 10^{11} neurónov prepojených navzájom pomocou synapsí. Neurón pracuje tak, že prijíma signál od iných neurónov, tento signál je na synapsii váhovaný a v tele neurónu sa prichádzajúce signály sčítajú. Ak ich hodnota presiahne určitý prah, daný neurón vyšle na svoj výstup impulz a tento sa cez synapsie dostane opäť k ďalším neurónom. Učenie neurónu predstavuje správne nastavovanie synaptických váh [29] [13].

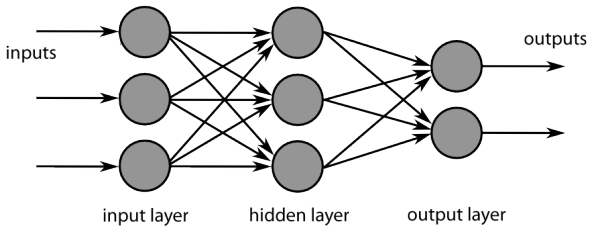
Na tomto princípe sú vystavané aj umelé neuróny. Sú navzájom prepojené a každý spoj je váhovaný, pričom učenie siete pozostáva práve z nastavovania týchto váh. Vo svojom tele obsahujú umelé neuróny bázové a aktivačné funkcie. Bázová funkcia slúži na zlúčenie vstupov z rôznych neurónov, pričom využíva už spomenuté váhy. Výsledok tejto bázovej funkcie následne slúži ako vstup aktivačnej funkcie, ktorá na základe určitej prahovej funkcie určí celkový výstup neurónu, ktorý môže byť 1 alebo 0 pri binárnych a 1 alebo -1 pri bipolárnych funkciách [29].

Neurónové siete môžu mať rôzne topológie, medzi hlavné patrí plne prepojená sieť (obrázok 3.4) a dopredná sieť (obrázok 3.5). Taktiež existuje niekoľko rôznych typov učenia,

pričom každá metóda má viacero predstaviteľov v podobe rôznych učiacich algoritmov. Rôznorodé sú aj možnosti využitia neurónových sietí, ktoré pozostávajú hlavne z klasifikácie a predikcie, ale napríklad aj zhlukovania alebo asociácie [29].



Obr. 3.4: Plne prepojená neurónová sieť.



Obr. 3.5: Dopredná neurónová sieť.

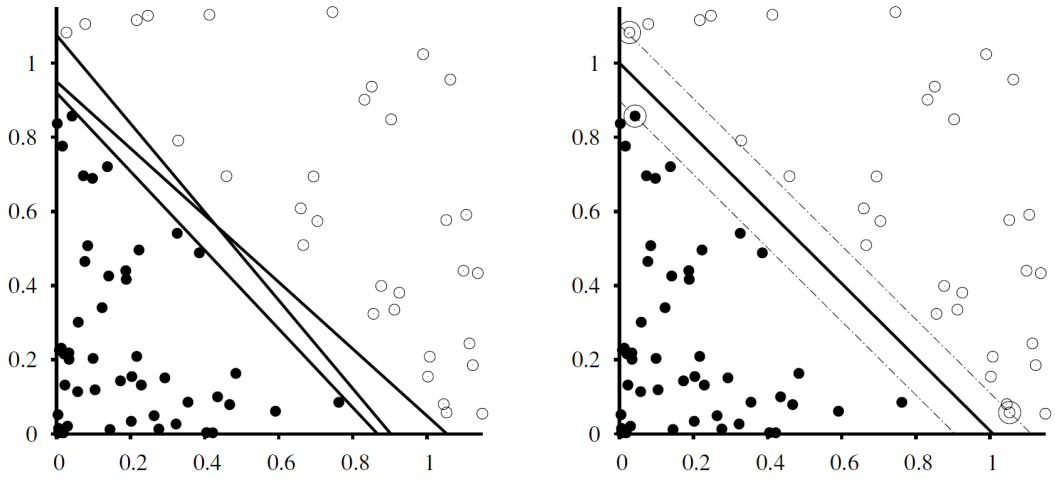
Ako vidno na obrázku 3.5, dopredné siete sa skladajú z vrstiev, pričom vstupná vrstva väčšinou len posúva vstupy ďalej, výstupná vrstva dáva na výstup finálne výsledky a skryté vrstvy (ktorých môže byť ľubovoľný počet, vrátane nuly) sa nachádzajú medzi vstupnou a výstupnou. V prípade absencie skrytej vrstvy sú vstupy spracovávané len jednou vrstvou a takáto sieť je schopná predstavovať len lineárne separabilné funkcie. S pridaním skrytej vrstvy už toto obmedzenie prestáva platiť a preto väčšina sietí využívaných v praxi obsahuje aspoň jednu skrytú vrstvu [29].

Neurónové siete sú mocný prístup strojového učenia a poskytujú často dobré výsledky, ich veľkou slabinou je však zlá interpretovateľnosť a čitateľnosť. Zatiaľ čo napríklad rozhodovacie stromy sú jasné a zrozumiteľné, neurónové siete tvoria akúsi čiernu skrinku a pohľad na vnútorné nastavenia váh jednotlivých neurónov a vnútorné hodnoty funkcii prináša pozorovateľovi takmer nulovú informáciu.

3.3.5 Support vector machines

Model *support vector machine* (ďalej len SVM) je hlavným predstaviteľom *metód s jadrom* a ide v podstate o prístup podobný už spomenutej lineárnej regresii a klasifikácii. Podobne ako logistická regresia sa snaží metóda SVM preložiť priestorom deliacu hyperplochu, ktorá by oddelovala vstupné dáta z odlišných tried. Rozdiel je v tom, že zatiaľ čo logistická regresia sa snaží minimalizovať počet chybne klasifikovaných vstupných vzoriek, SVM sa snaží

minimalizovať chyby zovšeobecňovania. To dosahuje tým, že volí deliacu hyperplochu tak, aby mala čo najväčšiu vzdialenosť od trénovacích príkladov. Tento rozdiel je badateľný na obrázku 3.6 [29].



Obr. 3.6: Rozdiel medzi lineárnym separátorom (a) a SVM (b) [29].

Vidíme, že v prvom prípade sú korektné všetky tri priamky separujúce od seba vstupy rôznych tried, ale mnoho vstupov je nebezpečne blízko pri priamke, čo vzbudzuje obavy, že nový vstupný bod sa ocitne na zlej strane. Oproti tomu SVM umiestni svoju deliacu priamku čo najďalej od všetkých vstupov, preto aj nové vstupy spadnú pravdepodobne do správnej výstupnej triedy, pokiaľ vychádzame z predpokladu, že ich rozloženie je rovnaké ako rozloženie trénovacích dát. Takýto separátor sa nazýva *separátor s maximálnym okrajom* (*maximum margin separator*), kde práve slovo *okraj* predstavuje vzdialenosť medzi čiarkovanými čiarami z obrázku 3.6(b).

Slová *support vector* v názve SVM označujú tie vstupné body, ktoré v podstate určujú výsledné umiestnenie deliacej hyperplochy (stávajú sa akousi *oporou* (*support*) deliacej hyperplochy). Na obrázku 3.6(b) sú to body v krúžku. Ide o body, nachádzajúce sa najbližšie pri deliacej hyperploche. Všetky body okrem *support vector* bodov majú nulové váhy a preto neovplyvňujú pozíciu deliacej hyperplochy.

Ďalším podstatným rysom SVM je, že dokážu rozdeliť aj skupiny bodov, ktoré nie sú lineárne separabilné. To docielia tým, že vstupné dáta namapujú z pôvodného N-dimenzionálneho priestoru do M-dimenzionálneho, pričom $M > N$. Pritom využívajú *jadrové funkcie* (*kernel functions*) na výpočet vektorových súčinov vstupných vektorov premapovaných do vyšších dimenzií, bez toho, aby SVM museli počítať dané mapovacie funkcie pre všetky vstupné body. Tento prístup sa nazýva *kernel trick* a vďaka nemu je možné efektívne nájsť optimálny lineárny separátor aj v priestoroch s miliardami dimenzií [29].

3.3.6 Konsenzuálne metódy

Konsenzuálne metódy (*ensemble methods*) môžeme považovať za akúsi nadstavbu nad ostatnými metódami strojového učenia. Nejde ani tak o novú metódu učenia, ako o zlúčenie

a využitie už existujúcich metód. Základná myšlienka konsenzuálnych metód je nepracovať s jediným predikčným (resp. klasifikačným) modelom, ale využiť niekoľko rôznych modelov, nad ktorými je následne možné vybudovať konsenzus, teda jednu výslednú predikciu.

Motivácia za týmto prístupom je nasledujúca: pri používaní jediného modelu môže dôjsť k nepresnostiam a chybám, ale pokiaľ využijeme viacero modelov, tak sa šanca chyby výrazne znižuje, pretože aby bol výsledný konsenzus chybný, musela by chybná predikcia nastať vo väčšine použitých modelov. Táto myšlienka platí hlavne v prípade, keď sú chyby jednotlivých modelov nezávislé. Pokiaľ by jednotlivé modely pracovali na rovnakom princípe, je možné predpokladať, že sa u všetkých modelov objaví rovnako chybná predpoveď a v tom prípade bude samozrejme chybný aj konsenzus [29]. Práve tento prístup bude použitý aj v tejto práci – cieľom je konsenzuálny prediktor, ktorý ako vstup prijíma výsledky rôznych existujúcich prediktorov a na základe metód strojového učenia zostaví konsenzus.

Kapitola 4

Dátová sada

Pre testovanie existujúcich prediktorov, ako aj pre trénovanie a testovanie vznikajúceho metaprediktoru, bolo nutné zozbierať dostatočné množstvo dát obsahujúcich sekvencie proteínov, ako aj záznamy o ich rozpustnosti. Prečítali sme články zaoberajúce sa predikciou rozpustnosti proteínov a sledovali, odkiaľ autori čerpali dáta. Následne sme z použiteľných zdrojov zozbierali potrebné dáta a spracovali ich.

4.1 Primárne zdroje dát

Autori rôznych článkov využívali na získanie dát rôzne metódy, od hľadania v literatúre cez vlastné experimenty až po využitie dát iných autorov. V tejto časti sa zameriame na vymenovanie databáz, použitých autormi článkov.

4.1.1 PubMed

Autori článku [18] využili bibliografickú databázu *PubMed*[1], kde pomocou kľúčových slov *soluble*, *inclusion bodies*, *E. coli* a *overexpression* vyhľadali proteíny, ktoré boli vytvorené prostredníctvom E. coli za normálnych podmienok, teda za normálnej teploty a bez prídavných látok na podporu expresie. Autori skontrolovali všetku literatúru, nájdenú vyhľadávaním so spomenutými kľúčovými slovami, a na základe toho získali zoznam rozpustných a nerozpustných proteínov. Podobne využili databázu aj autori článku [17] s použitím kľúčových slov *inclusion bodies*, *soluble*, *E. coli* a *overexpression*.

Takýto manuálny prístup je jednoznačne značne časovo náročný a autorom priniesol len malý počet proteínov (menej než 200 v prípade [18]). Z týchto dôvodov sme tento zdroj dát nevyužili.

4.1.2 eSOL

Databázu *eSOL*[2] využilo viacero autorov ([2], [26], [33], [30], [9]). Ide o databázu rozpustnosti celej sady E. coli proteínov, syntetizovaných pomocou *PURE*[3] systému, ktorý pracuje bez pomoci chaperonov. Experimenty s touto sadou boli vykonané na knižnici *ASKA*[4], ktorá pracuje s kmeňom K-12.

[1] http://www.ncbi.nlm.nih.gov/pubmed

[2] eSOL database(http://tp-esol.genes.nig.ac.jp/) developed in the Targeted Proteins Research Project. Databáza bola presunutá na adresu http://www.tanpaku.org/tp-esol/index.php?lang=en

[3] http://www.ncbi.nlm.nih.gov/pubmed/11479568

[4] http://www.ncbi.nlm.nih.gov/pubmed/16769691

Databáza obsahovala 4132 zázamov o proteínoch a ich rozpustnosti a využili sme ju pri tvorbe našej dátovej sady. Informácie o rozpustnosti sú vo forme percentuálnej rozpustnosti, čo našim potrebám nevyhovovalo. O konkrétnom postupe, ktorý musela databáza podstúpiť, sa bližšie zmieňujeme v časti 4.3.

4.1.3 TargetTrack

Databáza *TargetTrack*[5] vznikla zlúčením databáz *TargetDB* a *PepcDB*. Viacero autorov využilo jednu zo spomenutých troch databáz ([11], [31], [32], [24], [2]). Na stránke je možné vyhľadávať pomocou aminokyselinovej sekvencie, pomocou niektorých identifikátorov alebo podľa kľúčových slov. V každom prípade vyhľadávač nájde množinu záznamov, pri ktorých je k dispozícii okrem iného položka *Target summary*, ktorá v sebe obsahuje okrem iných aj položku *Progress*. Práve v tejto časti sa nachádzajú poznatky z experimentov s daným proteínom, z ktorých je možné vyčítať experimentálne overenú rozpustnosť či nerozpustnosť.

TargetTrack predstavuje bohatý zdroj dát, ktoré sme boli schopní využiť vďaka stránke prediktoru *ccSOL omics*[6]. Stránka okrem iných obsahuje dátovú sadu *TargetTrack*, rozdelenú na dve sady: rozpustnú a nerozpustnú, z ktorých každá obsahuje 18 495 proteínov. Autori dáta získali tak, že sledovali posledný dosiahnutý stav daného proteínu. Pokiaľ dosiahol daný proteín stav *soluble* alebo nejaký neskorší stav (napríklad *diffraction*, *crystal structure* alebo *in PDB*), považovali ho autori za rozpustný. Pokiaľ proteín nedosiahol stav *soluble* a zároveň bol označený kľúčovým slovom *work stopped*, považovali ho autori za nerozpustný. Tieto sady boli podľa autorov pozhlukované na úroveň 30% identity. Dátové sady sme využili pri tvorbe našej dátovej sady.

4.1.4 NCBI

Pri databázi *NCBI* sme narazili na podobný problém, ako pri databázi *PubMed*, konkrétne ten, že pre nájdené proteíny by bolo nutné manuálne prechádzať textové záznamy a vyberať relevantné. Túto databázu využívajú autori článku [37], neuvádzajú však spôsob, akým vybrali svoje dáta z databázy *NCBI*. Uvádzajú len, že vyhľadávali proteíny cez kľúčové slová *soluble* a *insoluble* a nájdené proteíny následne zredukovali na menšiu dátovú sadu, ktorú následne podrobili zhlukovaniu a ďalším úpravám. Nezistili sme, ako automatizovane vybrať relevantné záznamy z nájdených proteínov, preto sme tento zdroj nevyužili.

4.1.5 PDB

PDB[7] je proteínová databáza, ktorú využili autori článku [32]. Všetky proteíny obsiahnuté v *PDB* obsahujú okrem iného informácie o ich 3D štruktúre. Z databázy vybrali záznamy, ktorých anotácie obsahovali deskriptory *EXPRESSION_SYSTEM: ESCHERICHIA COLI* a *EXPRESSION_SYSTEM_VECTOR_TYPE: PLASMID*. Autori považujú všetky proteíny so známou 3D štruktúrou z princípu za rozpustné, preto všetky proteíny z *PDB* radia medzi rozpustné.

[5] http://sbkb.org/tt/
[6] http://s.tartaglialab.com/static_files/shared/tutorial_ccsol_omics.html#4
[7] http://www.rcsb.org/pdb/home/home.do

4.2 Sekundárne zdroje dát

Ako sme už uviedli, niektorí autori namiesto čerpania dát z databáz využili dáta, zozbierané inými autormi. V tejto časti sa zameriame práve na tieto dáta.

4.2.1 SOLP

Dátová sada *SOLP*[8] bola vytvorená autormi prediktoru *SOLpro* ([24]) a využili ju autori článku [20]. Autori dátovej sady využili pri jej tvorbe databázy *PDB* (4.1.5), *SwissProt*[9] a *TargetDB* (4.1.3). Okrem týchto primárnych zdrojov využili autori sady *SOLP* aj dátovú sadu zozbieranú autormi [18] (ktorí využili *PubMed*, sekcia 4.1.1).

Z databázy *PDB* získali autori proteíny exprimované prostredníctvom *E. coli* s využitím plazmidu ako vektoru. Následne vylúčili zo sady všetky proteíny, pri ktorých bolo pravdepodobné, že sú membránovými proteínmi. Túto skutočnosť autori buď vyčítali z anotácie proteínu, alebo využili program *TMHMM*[10], ktorý predikuje transmembránové špirály v proteínoch. Ďalej odstránili tie proteíny, ktorých sekvencie obsahovali dve a viac neznámych aminokyselín a nakoniec sekvencie, ktorých dĺžka vybočovala z rozsahu $< 10, 2000 >$.

Autori potom vybrali proteíny z databázy *SwissProt*, a to konkrétne tie, ktoré boli anotované kľúčovými slovami *E. coli*, *Enzyme* a *Reviewed*. Táto sada bola potom filtrovaná rovnako, ako sada z *PDB*.

TargetDB predstavoval jediný zdroj nerozpustných proteínov. Autori vybrali proteíny, označené ako *Cloned* alebo *Expressed*, ktoré zároveň neboli označené ako *Work stopped*. Z tejto sady vybrali tie, označené ako *Soluble* a pridali ich do sady rozpustných proteínov, a všetky ostatné proteíny z *TargetDB* sady pridali do sady nerozpustných proteínov. Potom výsledné sady filtrovali rovnako ako sady z *PDB* a *SwissProt*. Dáta z posledného zdroja ([18]) podstúpili rovnakú filtráciu.

Výsledné dátové sady boli zlúčené do jednej sady a pomocou programu *BLASTCLUST*[11] bola znížená redundancia s použitím hranice identity 25%. Následne boli náhodne odstraňované proteíny zo sady nerozpustných proteínov, aby sa dosiahol rovnaký počet záznamov v oboch sadách. Výsledná dátová sada obsahuje 17 408 proteínov, z ktorých polovica je rozpustná a polovica nerozpustná. Túto dátovú sadu sme využili pri tvorbe našej finálnej dátovej sady.

4.2.2 PROSO

Sada *PROSO*[12] bola vytvorená autormi prediktorou *PROSO II* ([31]) a využíva databázu *pepcDB* (dnes už *TargetTrack*, viď 4.1.3). Autori sledovali experimentálne výsledky a históriu experimentov, uchovávanú pre každú sekvenciu, a na základe stavu experimentu rozdelili sekvencie na rozpustné a nerozpustné. Sekvencie, ktoré dosiahli aspoň stavu *Soluble* alebo nejakej neskoršej fázy (napr. *Purified*) označili za rozpustné. Sekvencie, ktoré po dobu 8 mesiacov (od septembra 2009 do mája 2010) zotrvali vo fázach nižších než *Soluble* označili autori za nerozpustné. Dodatočne porovnali sadu nerozpustných proteínov s databázou *PDB* a pokiaľ táto databáza obsahovala 100% zhodu s niektorým proteínom, označeným ako nerozpustný, tento proteín bol zo sady odstránený.

[8] http://download.igb.uci.edu/
[9] http://www.uniprot.org/uniprot/
[10] http://www.cbs.dtu.dk/services/TMHMM/
[11] http://www.ncbi.nlm.nih.gov/Web/Newsltr/Spring04/blastlab.html
[12] http://mips.helmholtz-muenchen.de/prosoII/img/Suppl_files.zip

Je nutné podotknúť, že mnoho proteínov z databázy *pepcDB* neobsahovalo opis expresného systému, ale autori uviedli, že viac než 75% proteínov, ktoré dosiahnu stav *in PDB* je produkovaných prostedníctvom *E. coli*. Podobne ako v prípade sady *SOLP* (4.2.1) vylúčili autori proteíny, obsahujúce aspoň jeden transmembránový segment. Túto vlastnosť rovnako ako autori sady *SOLP* predikovali s využitím nástroja *TMHMM*. Taktiež vylúčili proteíny s dĺžkou menšou než 20, prípadne väčšou než 2004 aminokyselín.

V poslednom kroku bola odstránená redundancia pomocou zhlukovania na podobnosť 90%, 75%, 50% a 25%, na čo autori využili program *CD-HIT*[13]. Rôzne úrovne zhlukovania využili autori pri tvorbe svojho prediktoru a pre 90% úroveň identity zdôrazňujú veľkosť vytvorenej dátovej sady, ktorú tvorí 82 299 proteínov. Autori neuviedli, ktorá úroveň zhlukovania bola vykonaná na dátach, poskytnutých verejnosti, ale podľa počtu týcho dát (približne 130 000) sme usúdili, že autori zverejnili pôvodnú, nezhlukovanú dátovú sadu. Túto dátovú sadu sme využili pri tvorbe našej dátovej sady.

4.2.3 Niwa et al.

Na stránke prediktoru *ccSOL omics* sa nachádza niekoľko dátových sád od rôznych autorov, medzi nimi aj dátová sada vytvorená autormi článku [26]. Táto sada bola vytvorená experimentami nad knižnicou *ASKA* a obsahuje tak výhradne proteíny z E. coli. Autori vytvorili 3 173 proteínov, ktorým pomocou centrifúgy experimentálne priradili mieru rozpustnosti.

Konkrétne tento proces prebiehal nasledovne: autori amplifikovali ORF rámce z *ASKA* knižnice, tieto boli následne preložené systémom bez chaperonov. Po preklade zobrali autori časť výsledného roztoku a podrobili ju centrifugácii. Pomer časti, ktorá po centrifugácii zostala oddelená od zvyšku, k časti, ktorá nebola podrobená centrifugácii, považovali autori za hodnotu rozpustnosti. Sada zo stránky prediktoru *ccSOL omics* obsahuje spolu 2 159 proteínov a bola taktiež použitá pri tvorbe našej výslednej dátovej sady.

4.3 Spracovanie dát

Pre našu dátovú sadu sme zozbierali dáta zo siedmych zdrojov, uvedených v predchádzajúcej časti. Väčšina dát bola vo vyhovujúcom formáte, konkrétne v textových súboroch obsahujúcich aminokyselinové sekvencie a binárny stav rozpustnosti: rozpustný alebo nerozpustný. Výnimku tvorila dátová sada *eSOL*, ktorej sa venujeme v časti 4.3.1.

4.3.1 Spracovanie dátovej sady *eSOL*

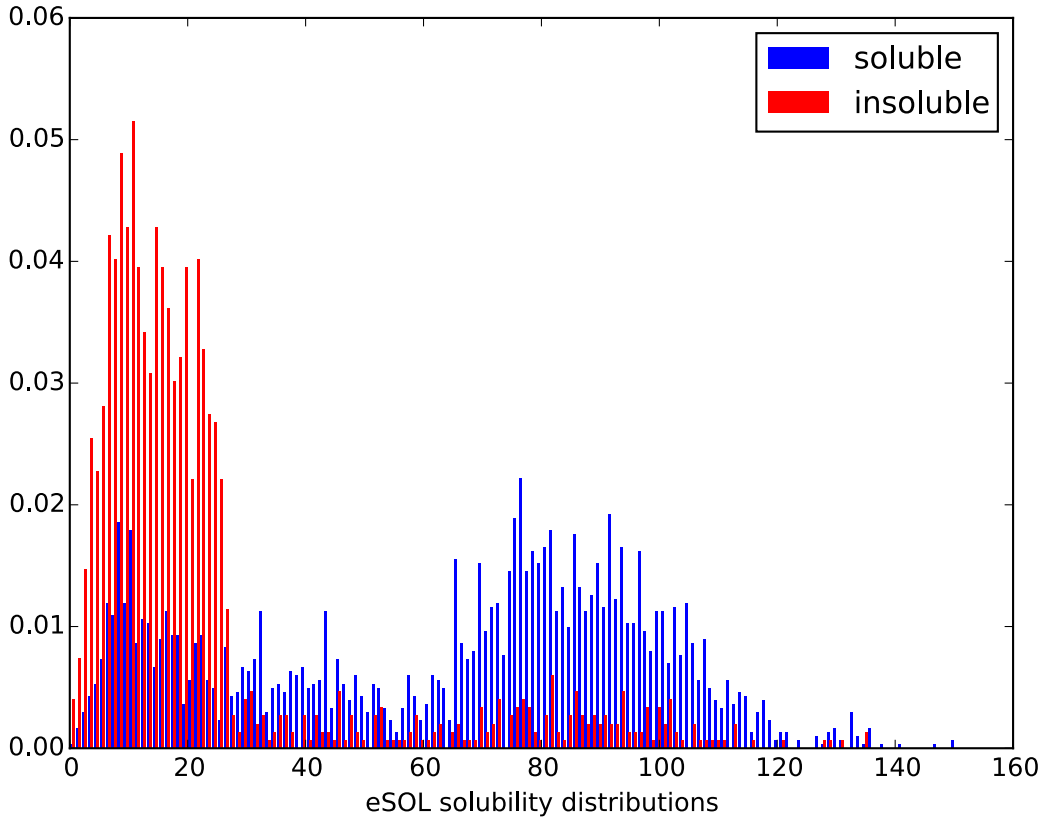
Dátová sada mala dve nepriaznivé vlastnosti: jednak neobsahovala aminokyselinové sekvencie, a jednak namiesto binárnej hodnoty rozpustnosti obsahovala percentuálnu rozpustnosť. Problém s absenciou sekvencií sme vyriešili tým, že sme využili atribút *ECK number*[14]. S využitím stránky *ecogene*[15] sme stiahli databázu a ako atribúty sme zvolili okrem iného práve ECK číslo a *SwissProt* ID. Cez ECK čísla sme k jednotlivým proteínom z *eSOL* databázy priradili *SwissProt* ID, následne sme stiahli databázu proteínov zo *SwissProt*, ktorá obsahovala aj aminokyselinové sekvencie, a tak konečne spojili záznamy z databázy *eSOL* so sekvenciami.

[13] http://weizhongli-lab.org/cd-hit/

[14] ECK numbers were assigned to the E. coli K-12 MG1655 and W3110 genomes in 2005 in an attempt to provide shared accession numbers for genes common to the two genomes [36].

[15] http://www.ecogene.org/ecodownload/dbtable

Problém s binárnou hodnotou rozpustnosti sme vyriešili porovnaním sekvencií z *eSOL* so sekvenciami zvyšných dátových sád, ktoré obsahovali binárnu hodnotu rozpustnosti. Vzhľadom na istý prekryv databázy *eSOL* s ostatnými sme takto získali sekvencie, pri ktorých bola uvedená rozpustnosť ako percentuálne, tak aj binárne. Spočítali sme distribúciu percentuálnych hodnôt rozpustnosti pre binárne nerozpustné a binárne rozpustné proteíny, výsledok je na obrázku 4.1.



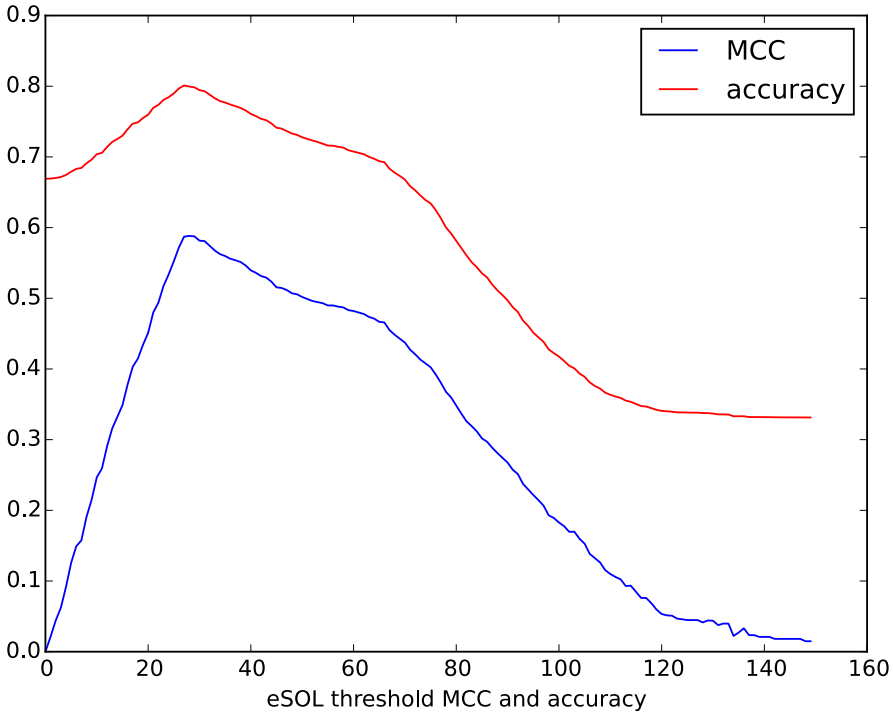
Obr. 4.1: Distribúcia percentuálnych hodnôt rozpustnosti sady eSOL.

Vidíme, že nerozpustné proteíny sa pohybujú prevažne pod hranicou 30%, zatiaľ čo rozpustné proteíny sú viac rozložené. Následne sme posúvali hranicu rozpustnosti od 0% do 150% (pretože niektoré proteíny v dátovej sade *eSOL* boli ohodnotené percentuálnou rozpustnosťou nad 100%) a pre každú hodnotu hranice sme spočítali presnosť rozdelenia, ako aj MCC (*Matthews correlation coefficient*). Výsledok je znázornený na obrázku 4.2.

Na základe týchto experimentov sme zvolili hranicu rozpustnosti 30%, a všetky proteíny, ktoré túto hranicu dosiahli alebo prekročili, sme označili za rozpustné, ostatné za nerozpustné.

4.3.2 Tvorba výslednej dátovej sady

Po vyriešení problémov s dátovou sadou *eSOL* už boli všetky dátové sady vo vhodnom formáte. Všetky dátové sady sme teda zlúčili do jednej, čím sme získali dátovú sadu s 222 041 proteínmi. Spočítali sme prieniky dvojíc dátových sád a výsledok zobrazujeme v tabuľke 4.1.



Obr. 4.2: Vhodnosť nastavenia prahu percentuálnej rozpustnosti sady eSOL.

Vidíme, že dátová sada *PROSO_c* je podmnožinou dátovej sady *PROSO*, a rovnako je sada *SOLP_c* podmnožinou sady *SOLP*. Taktiež vidno, že dátová sada *Niwa et al.* je takmer celá zložená zo sekvencií zo sady *eSOL*, čo nie je prekvapením, pretože autori sady *Niwa et al.* pracovali práve so sadou *eSOL*. Medzi hlavné dátové sady teda nakoniec patria sady *eSOL*, *PROSO*, *SOLP* a *TargetTrack* a ich prekryvy dodatočne ukazuje obrázok 4.3. Pri počítaní prienikov dát sme narazili na situácie, kde jedna dátová sada obsahovala jednu sekvenciu viackrát a niekedy dokonca s rôznou hodnotou rozpustnosti. Tabuľka 4.2 ukazuje naše zistenia.

Spočítali sme taktiež kolízie v celých dátach, teda prípady, keď je rovnaká sekvencia anotovaná rôznymi hodnotami rozpustnosti, nezávisle od dátovej sady. Zhodnotili sme, že keď aspoň jeden experiment nad danou sekvenciou spôsobil kladnú anotáciu rozpustnosti, môžeme danú sekvenciu považovať za rozpustnú. V dátach sme našli 15 793 takýchto kolíznych dvojíc a dané proteíny sme teda označili za rozpustné.

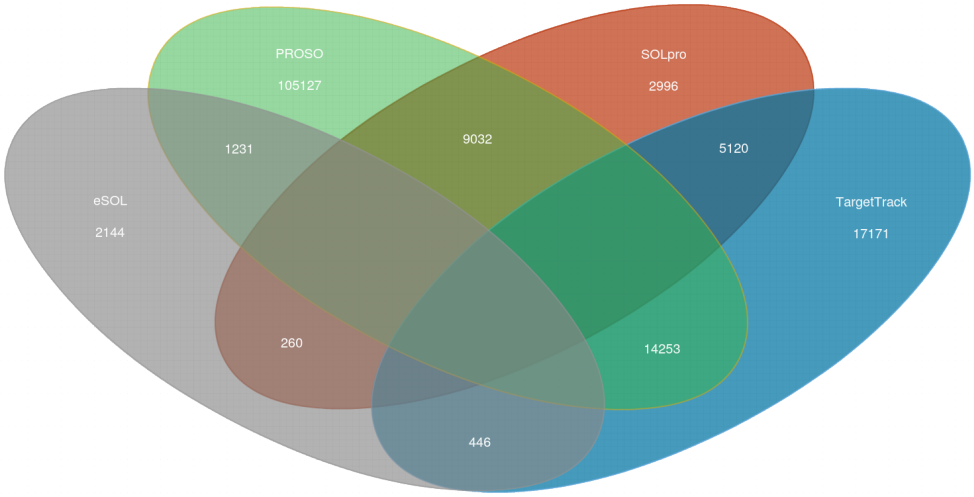
Pri spôsobe zberu dát, ktorý sme využili, dochádza k výraznej redundancii. Preto sme použili program *CD-HIT* [22][10] a prostredníctvom organizácie *MetaCentrum*[16] sme dáta zhlukovali na úrovne 90%, 80%, 70%, 60%, 50% a 40% identity sekvencií. Dodatočne sme dáta pozhlukovali ešte na 30% identitu s využitím programu *PSI-CD-HIT*, ktorý je súčasťou balíčku programov *CD-HIT*. Výsledky zhlukovania s konkrétnymi počtami zobrazuje tabuľka 4.3.

Z výsledných 52 828 je 22 557 proteínov rozpustných a 30 271 nerozpustných. Nie je žiaduce, aby pri testovaní prediktorov prvej úrovne boli použité sekvencie, na ktorých autori daných prediktorov tieto prediktory trénovali. Vieme, z akého zdroja ktorá sekvencia našej

[16] https://metavo.metacentrum.cz/

	eSOL	Niwa et al.	PROSO_c	PROSO	SOLP	SOLP_c	TargetTrack
eSOL	**4 081**	-	-	-	-	-	-
Niwa et al.	2 141	**2 159**	-	-	-	-	-
PROSO_c	277	178	**22 614**	-	-	-	-
PROSO	1 231	781	22 614	**129 643**	-	-	-
SOLP	260	147	6 689	9 032	**17 408**	-	-
SOLP_c	160	96	7 272	7 112	9 146	**9 146**	-
TargetTrack	446	182	9 803	14 253	5 120	4 907	**36 990**

Tabuľka 4.1: Označenia *PROSO_c* a *SOLP_c* predstavujú dátové sady PROSO a SOLP, získané zo stránok prediktoru *ccSOL omics*. Hlavná diagonála ukazuje počty sekvencií v jednotlivých dátových sadách. Políčka pod diagonálou ukazujú počet sekvencií, spoločných pre jednotlivé dvojice dátových sád.



Obr. 4.3: Prienik hlavných dátových sád.

dátovej sady pochádza, a môžeme teda pre konkrétny prediktor prvej úrovne vylúčiť konkrétne sekvencie. Problémom však je, že pri zhlukovaní sa viac sekvencií, ktoré môžu mať pôvod v rôznych dátových sadách, zlúči do jednej, a tým sa stráca informácia o pôvode danej sekvencie. Na vyriešenie tohto problému si uchovávame pri každej sekvencii aj jej tzv. *alternatívne ID*, čo sú identifikátory, ktoré niesli všetky sekvencie, ktoré boli zhlukovaním zlúčené s jedinou výslednou sekvenciou.

Ako príklad si môžeme predstaviť 3 sekvencie, jednu s ID 1 pochádzajúcu z dátovej sady prediktoru *PROSO II*, druhú s ID 2, pochádzajúcu z dátovej sady prediktoru *SOLpro* a tretiu s ID 3, pochádzajúcu z databázy *TargetTrack*. Povedzme, že pri zhlukovaní sú sekvencie 2 a 3 zlúčené so sekvenciou 1. Pre sekvenciu 1 sa teda uchovajú alternatívne ID 2 a 3. Keď budeme následne chcieť testovať prediktor *SOLpro*, sekvenciu s ID 1 nepoužijeme, pretože medzi jej alternatívnymi ID sa nachádza ID patriace medzi sekvencie dátovej sady *SOLpro*.

Ako alternatívu k alternatívnym ID sme do databázy zakomponovali vzťah, kde každý proteín patrí do jednej alebo viacerých pôvodných dátových sád. Používateľ databázy tak

	eSOL	Niwa et al.	PROSO_c	PROSO	SOLP	SOLP_c	TargetTrack
Zhodné	9	0	0	0	338	0	0
Nezhodné	8	0	920	5	0	399	0

Tabuľka 4.2: Tabuľka ukazuje počty duplicitných sekvencií v dátových sadách. Prvý riadok ukazuje duplicity, pri ktorých je však zapísaná rovnaká hodnota rozpustnosti. Druhý riadok ukazuje tie duplicity, kde majú rovnaké sekvencie zapísanú rôznu rozpustnosť.

Nezhlukované	90%	80%	70%	60%	50%	40%	30%
222 041	103 085	98 786	94 595	89 035	81 062	68 907	52 828

Tabuľka 4.3: Počty sekvencií pri zhlukovaní.

môže priamo získať tie proteíny, ktoré patria (buď priamo, alebo cez alternatívne ID) do konkrétnej pôvodnej dátovej sady.

Tabuľka 4.4 ukazuje, aké veľké časti výslednej dátovej sady boli použité pri trénovaní alebo testovaní konkrétnych nástrojov.

Nástroj	Počet dát - priamo	Počet dát - altID	Počet použiteľných dát
Davis et al.	-	-	52 828 (100%)
SOLpro	1 219 (2,31%)	15 449 (29,24%)	36 160 (68,45%)
PROSO II	32 828 (62,14%)	4 731 (8,96%)	15 253 (28,87%)
ESPRESSO	-	-	52 595 (99,56%)

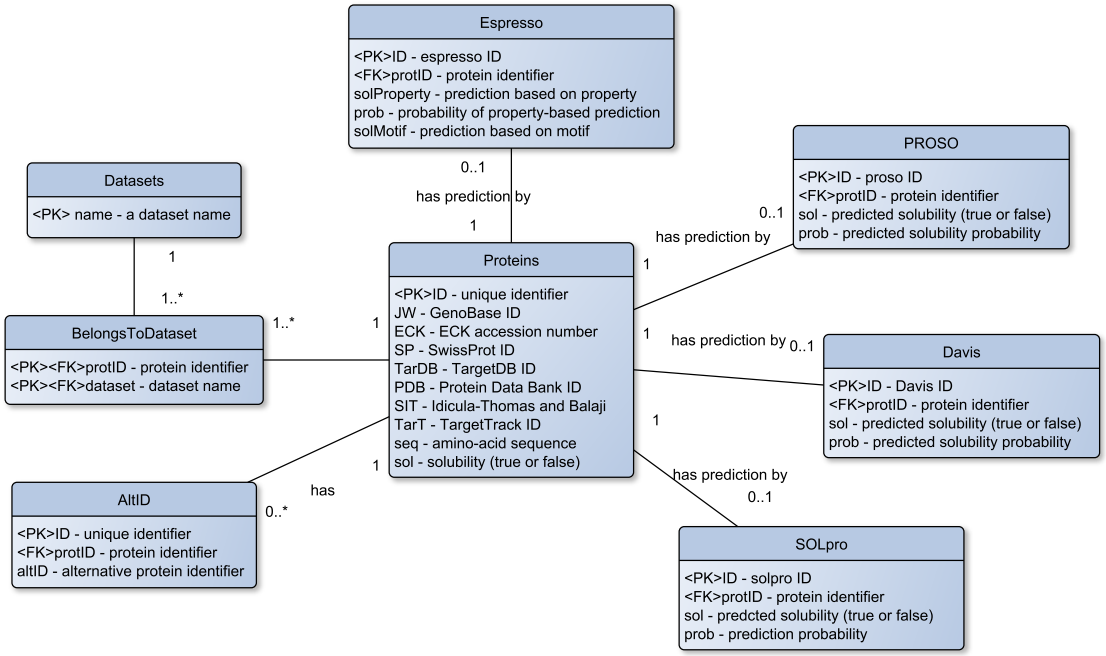
Tabuľka 4.4: Prekryv dátovej sady so sadami nástrojov. Druhý stĺpec predstavuje počet sekvencií, ktoré patria priamo do dátovej sady niektorého nástroja. Tretí stĺpec predstavuje počet tých sekvencií, ktoré priamo nepatria do dátovej sady daného nástroja, ale aspoň jedno z ich alternatívnych ID patrí do dátovej sady daného nástroja. Posledný stĺpec predstavuje počet dát, ku ktorým sme získali predikciu od daného nástroja. Väčšinou je to zvyšok dát po odčítaní počtov z druhého a tretieho stĺpca, ale môže byť menší, pokiaľ má nástroj nejaké špeciálne požiadavky na sekvencie (napr. absenciu neznámych aminokyselín). Autori nástrojov *Davis et al.* a *ESPRESSO* nezverejnili konkrétne trénovacie dáta, preto nepoznáme prekryv s našou sadou a teda hodnoty v poslednom stĺpci je nutné brať s istou rezervou.

4.3.3 Uloženie dát

Po zistení, ktoré prediktory rozpustnosti budeme používať, sme vytvorili schému uloženia dát do našej databázy. O samotných prediktoroch sa bližšie zmieňujeme v časti 5. Pre tvorbu databázy sme využili program *MySQL*[17] a vytvorili sme si lokálnu databázu, ktorej schéma je znázornená na obrázku 4.4.

Základná tabuľka *Proteins* obsahuje všetky proteíny, ktoré sú výsledkom zhlukovania. Uchováva rôzne identifikátory, ktoré sme získali z rôznych zdrojov, pričom nie všetky

[17] https://www.mysql.com/



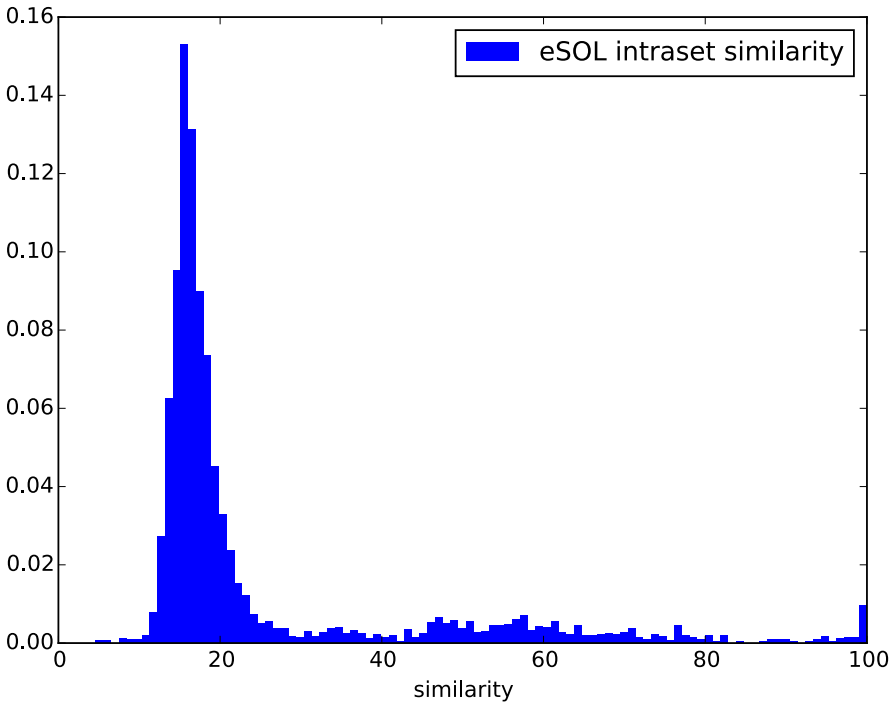
Obr. 4.4: Schéma databázy.

musia byť vždy uvedené. Hlavným identifikátorom je naše vlastné unikátne ID, nezávislé od rôznych získaných identifikátorov. Tabuľka *AltID* obsahuje už zmienené alternatívne ID a odkazuje sa svojím cudzím kľúčom na naše vnútorné ID. Tabuľka *Datasets* obsahuje názvy pôvodných dátových sád, v našej práci teda obsahuje 7 položiek. Keďže išlo pri vzťahu proteínov a dátových sád o problém M ku N, bolo nutné vytvoriť väzobnú tabuľku *BelongsToDataset*. Tabuľky *Espresso*, *PROSO*, *SOLpro* a *Davis* obsahujú predikovanú rozpustnosť a pravdepodobnosť tejto predikcie, a svojím cudzím kľúčom sa taktiež odkazujú na naše vnútorné ID z tabuľky *Proteins*.

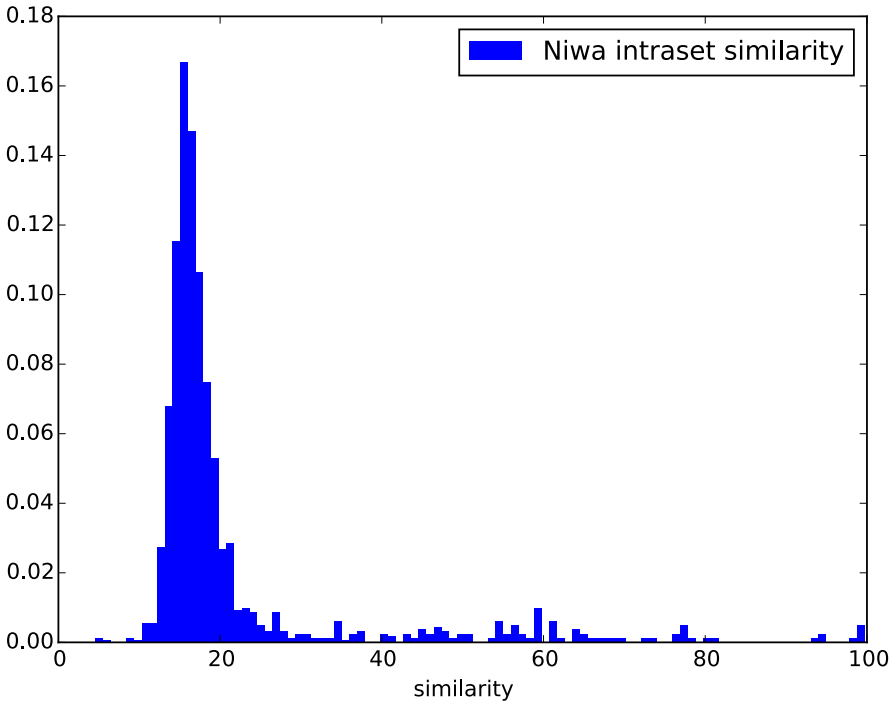
4.3.4 Analýza dát

Vykonali sme analýzu podobnosti sekvencií v rámci jednotlivých dátových sád a výsledky ukazujeme na obrázkoch 4.5, 4.6, 4.7, 4.8, 4.9, 4.10 a 4.11 v podobe histogramov podobností dvojíc sekvencií. Tieto podobnosti boli merané na pôvodných, nezhlukovaných dátových sadách. Meranie podobnosti prebiehalo na základe semiglobálneho zarovnania, takže stopercentné podobnosti v grafoch nemusia nutne znamenať identické sekvencie, ale môžu predstavovať kratšie sekvencie, ktoré sa celé nachádzajú v niektorej z dlhších sekvencií.

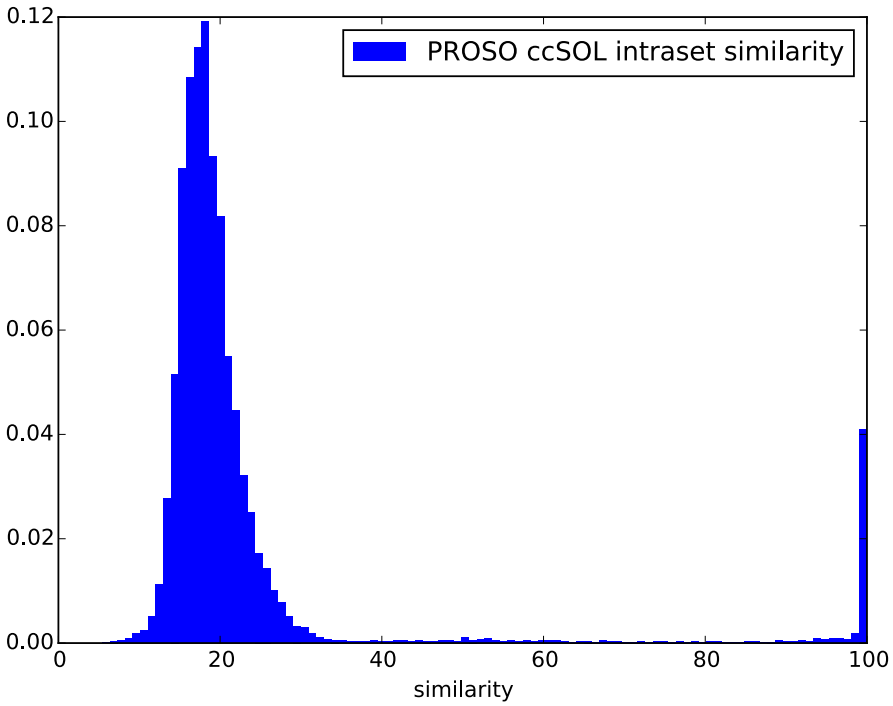
Väčšina sád ukazuje mieru identity okolo hranice 20%, pričom niektoré vykazujú nárast aj v oblasti 100%, čo poukazuje na redundanciu v rámci niektorých sád. Je nutné upozorniť, že pri získavaní dvojíc podobných sekvencií sme vyberali pre každú sekvenciu len 10 jej najpodobnejších, takže pokiaľ nejaká sekvencia mala v danej dátovej sade viac než 10 podobných sekvencií, vybrali sme tie s najvyššími podobnosťami. V takýchto prípadoch je možné, že sú ukázané grafy viac naklonené k vyšším hodnotám miery identity, než aké sú skutočne prítomné v dátach. Rovnakú analýzu sme vykonali aj nad našou výslednou dátovou sadou a výsledok vidno na obrázku 4.12.



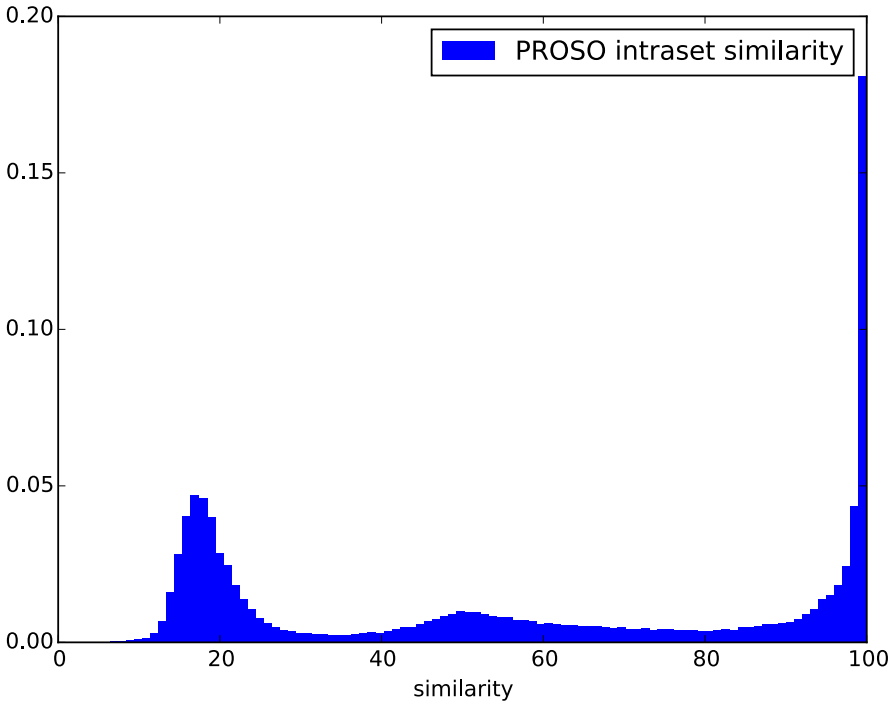
Obr. 4.5: Podobnosť dvojíc sekvencií v rámci sady eSOL.



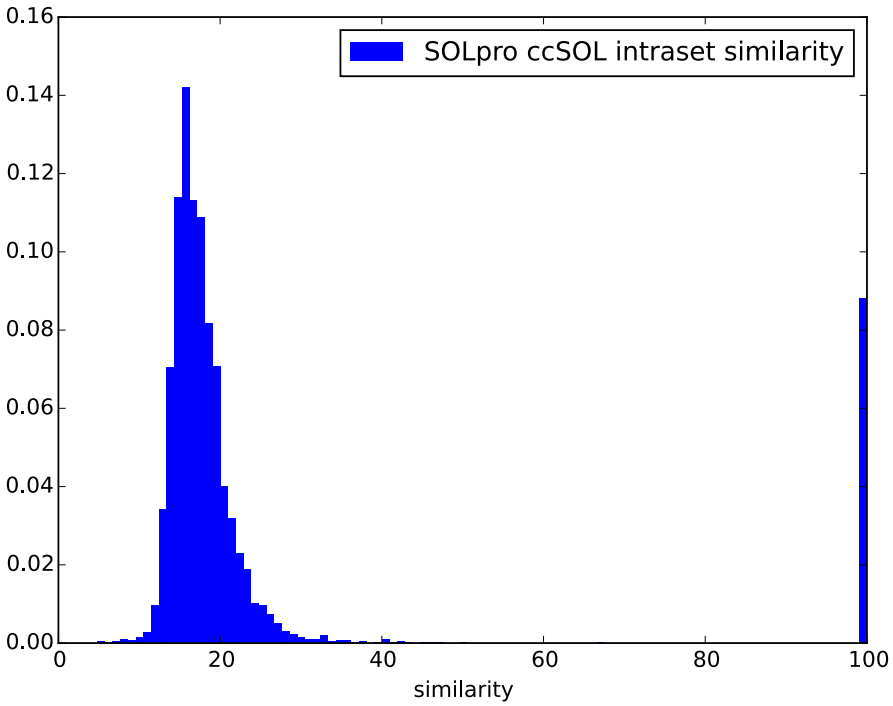
Obr. 4.6: Podobnosť dvojíc sekvencií v rámci sady Niwa et al.



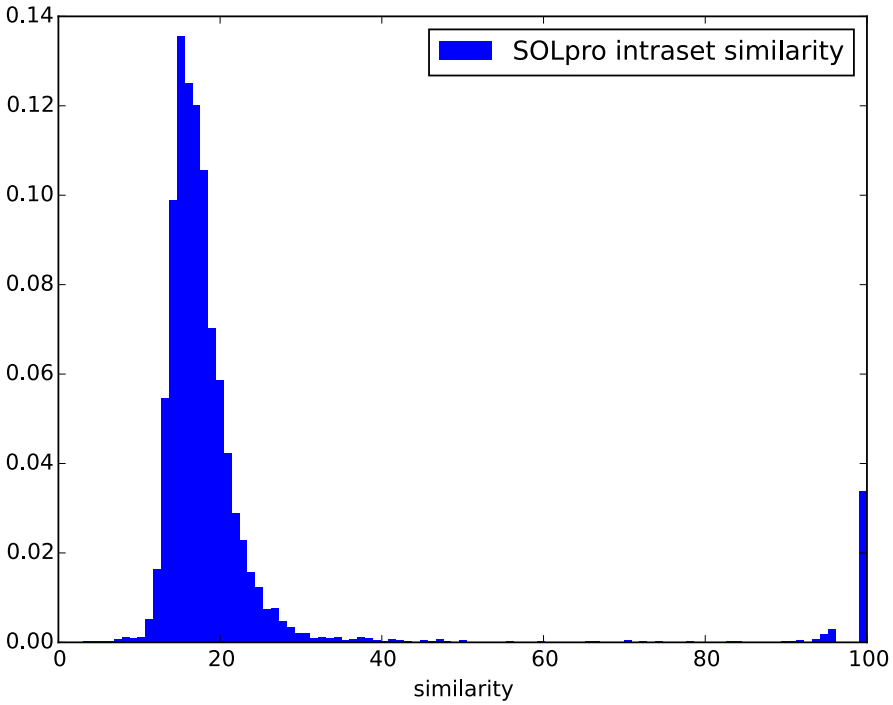
Obr. 4.7: Podobnosť dvojíc sekvencií v rámci sady PROSO ccSOL.



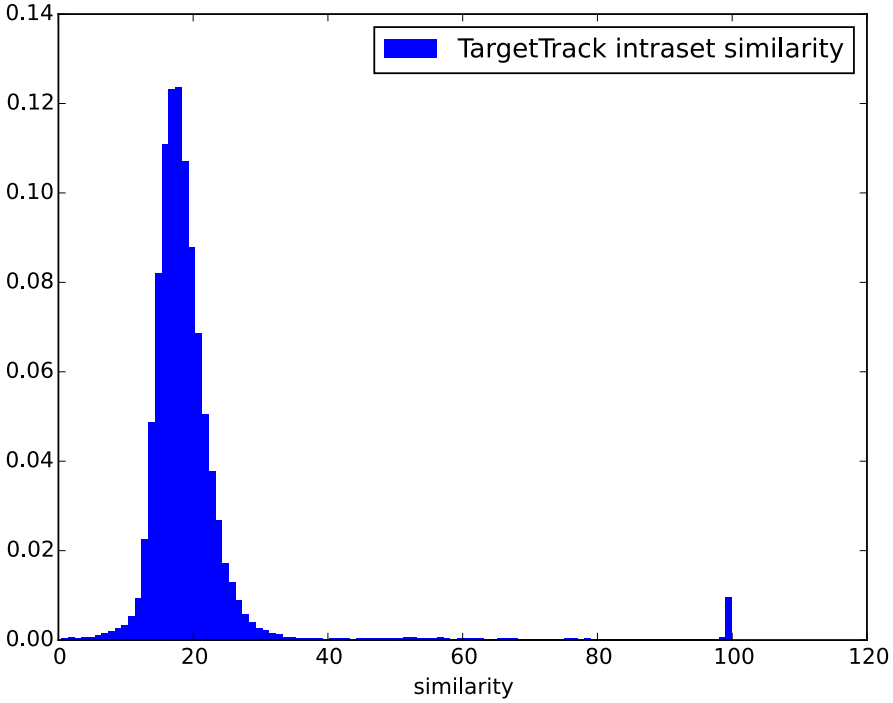
Obr. 4.8: Podobnosť dvojíc sekvencií v rámci sady PROSO.



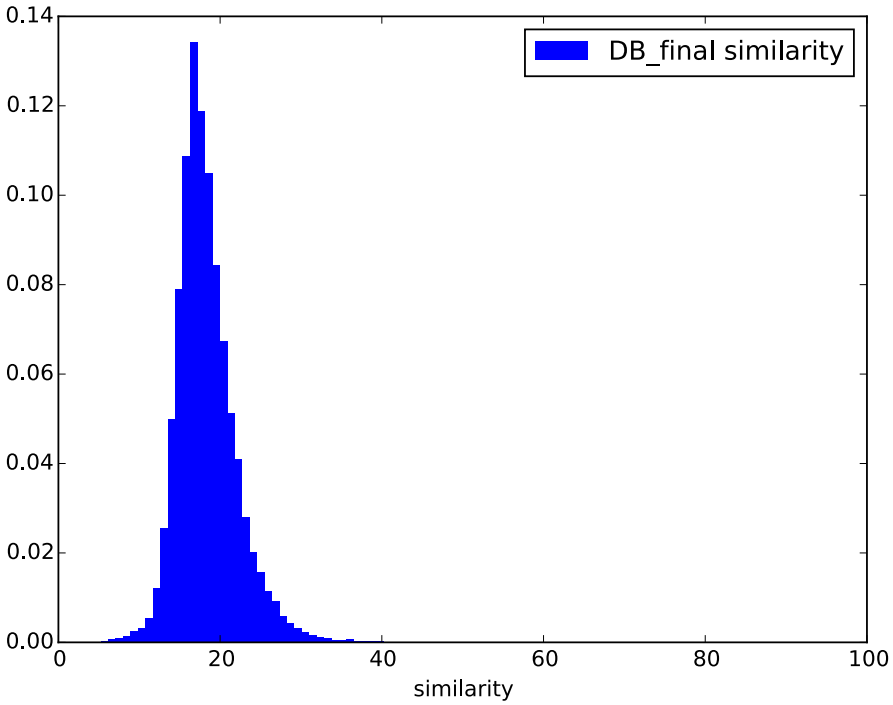
Obr. 4.9: Podobnosť dvojíc sekvencií v rámci sady SOLpro ccSOL.



Obr. 4.10: Podobnosť dvojíc sekvencií v rámci sady SOLpro.



Obr. 4.11: Podobnosť dvojíc sekvencií v rámci sady TargetTrack.



Obr. 4.12: Podobnosť dvojíc sekvencií v rámci výslednej dátovej sady.

Kapitola 5

Prediktory

Otázka predikcie rozpustnosti proteínov je stále aktuálna, avšak nejde o nový problém. Z tohto dôvodu už vzniklo mnoho prediktorov, ktoré sa pomocou rôznych techník strojového učenia snažili predvídať rozpustnosť či nerozpustnosť proteínov. V časti 5.1 všeobecne opisujeme všetky prediktory, ktoré sme skúmali a následne v časti 5.2 sa detailnejšie venujeme prediktorom, ktoré sme využili v našej práci.

Niektorí autori používajú pri predikcii vlastné abecedy aminokyselín, čo znamená, že zhlukujú viaceré aminokyseliny, ktoré z určitého pohľadu považujú za rovnaké, pod jediný symbol. Takáto vlastná abeceda bude v texte značená nasledovne:

- **vlastnosť**: [CMQ], [AG].

Tento príklad ukazuje, že aminokyseliny C, M a Q majú istú spoločnú vlastnosť, a pokiaľ sa v sekvencii vyskytne ktorákoľvek z týchto troch aminokyselín, zvýši sa počítadlo výskytov symbolu [CMQ]. Podobne aminokyseliny A a G majú spoločnú danú vlastnosť, ale odlišnú od kategórie [CMQ].

Presnosti prediktorov, ktoré nižšie predstavujeme, je možné chápať v zmysle *accuracy* a sú teda počítané podľa vzorca:

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.1}$$

kde symboly TP, TN, FP a FN znamenajú *true positive*, *true negative*, *false positive* a *false negative*.

5.1 Prehľad prediktorov

Pri hľadaní použiteľných prediktorov sme vychádzali z článku [12], ktorý zhŕňa existujúce prediktory a odkazuje na články, ktoré o jednotlivých prediktoroch pojednávajú. Po prečítaní jednotlivých článkov sme žiaľ zistili, že mnoho prediktorov vzniklo len pre účely výskumu, ktorému sa daný článok venoval, a teda nie sú vôbec dostupné.

Pokiaľ sme daný prediktor sami použili, uvádzame aj nami nameranú presnosť. Táto presnosť vychádza z testovania na našej dátovej sade, a to konkrétne na najväčšej možnej podmnožine tak, aby daná podmnožina neobsahovala sekvencie, ktoré autori daného prediktoru použili pri jeho trénovaní, a aby zároveň obsahovala rovnaký počet rozpustných a nerozpustných proteínov.

5.1.1 Davis et al.

Princíp predikcie

Autormi navrhnutý model ([7]) pracuje na princípe regresie, využíva teda rovnicu, do ktorej je nutné dosadiť parametre, vyčítané z aminokyselinovej sekvencie proteínu. Tento model je modifikovanou verziou modelu od pánov Wilkinsona a Harrisona ([35]). Narozdiel od verzie z roku 1991 využíva model pána Davisa len dva z pôvodných piatich rysov, a to *priemerný celkový náboj* (počítaný ako rozdiel množstva kyseliny aspartánovej a glutámovej oproti lyzínu a arginínu) a *obsah aminokyselín tvoriacich otočky*[1] (počítaný ako celkový počet asparagínu, glycínu, prolínu a serínu).

Konkrétna podoba použitej rovnice je:

$$CV = \lambda_1 \left(\frac{N + G + P + S}{n} \right) + \lambda_2 \left| \frac{(R + K) - (D + E)}{n} - 0.03 \right| \tag{5.2}$$

kde n je celkový počet aminokyselín v proteíne, N, G, P, S, R, K, D, E sú počty konkrétnych aminokyselín a λ_1 a λ_2 sú koeficienty s hodnotami 15.43 a -29.56. Získaná hodnota CV sa následne použije na výpočet rozdielu $CV - CV'$, kde CV' má hodnotu 1.71. Ak je tento rozdiel kladný, proteín bude predikovaný ako nerozpustný, inak bude predikovaný ako rozpustný. Pravdepodobnosť získanej predikcie sa počíta podľa vzorca:

$$0.4934 + 0.276|CV - CV'| - 0.0392(CV - CV')^2 \tag{5.3}$$

Trénovanie a testovanie

Článok sa výnimočne nezaoberá predikciou ako takou, ale pojednáva o fúznych systémoch pri produkcii rekombinantných proteínov. Z tohto dôvodu nie je možné hovoriť o trénovaní a testovaní a ani samotný článok sa o týchto fázach nijak nezmieňuje.

Zhrnutie

Autori vzhľadom na povahu článku neuvádzajú presnosť prediktoru. Vzhľadom na jednoduchosť implementácie regresie a zverejnenú rovnicu sme si tento prediktor sami naprogramovali a pri testovaní nad našou dátovou sadou dosiahol presnosť približne 52%. V prípade tohto prediktoru bol počet testovacích vzoriek dvakrát 22 557. Nami nameraná presnosť sa blíži presnostiam, ktoré uviedli autori článkov [24] (54%) a [15] (60%), ktorí tento model taktiež testovali.

5.1.2 Idicula-Thomas et al. 2005

Princíp predikcie

Autori ich navrhnutý systém ([18]) nezamýšľali ako prediktor, ale ako overenie korelácie medzi niektorými sekvenčnými vlastnosťami proteínov a ich rozpustnosťou. Tento systém spočíva vo výpočte *indexu rozpustnosti* (angl. *solubility index*), ktorého rovnica nasleduje.

[1] turn-forming residue content

$$SI = \frac{0,648 \cdot AI + 0,274 \cdot II_N - 0,539 \cdot F_N - 0,508 \cdot F_T - 0,604 \cdot F_Y + S_{TP} \cdot 10^4}{100} \quad (5.4)$$

AI predstavuje alifatický index skúmaného proteínu, II_N je index nestability N-konca (*instability index of the N-terminus*), F_N, F_T a F_Y sú frekvencie výskytu asparagínu, treonínu a tyrozínu a S_{TP} je takzvané *tripeptidové skóre* (*tripeptide score*). Pokiaľ výsledok rovnice prekročí pre zadaný proteín hodnotu 1, autori predpokladajú vysokú pravdepodobnosť rozpustnosti proteínu. V opačnom prípade dávajú autori proteínu nízku pravdepodobnosť rozpustnosti.

Alifatický index priamo odpovedá množstvu alanínu, valínu, leucínu a izoleucínu a počíta sa nasledovne:

$$AI = X(Ala) + 2,9 \cdot X(Val) + 3,9 \cdot [X(Ile) + X(Leu)] \quad (5.5)$$

kde X predstavuje mólové percento zložky v zátvorke. Index nestability súvisí so stabilitou proteínu a počíta sa podľa nasledujúcej rovnice:

$$II = \frac{10}{L} \cdot \sum_{i=1}^{L-1} DIWV(x_i y_{i+1}) \quad (5.6)$$

kde L je počet aminokyselín v danom proteíne a $DIWV$ je *instability weight value* pre dipeptid $x_i y_{i+1}$, autori však neuvádzajú, čo táto hodnota znamená, ani ako sa počíta. V rovnici 5.4 pracujú autori s indexom nestability N-konca, kde namiesto všetkých aminokyselín pracujú len s aminokyselinami N-konca. V tejto práci autori definovali N-koniec ako prvých 20 aminokyselín.

Poslednou časťou je tripeptidové skóre, počítané podľa nasledujúceho vzorca:

$$S_{TP} = \frac{1}{L-2} \cdot \sum_{i=0}^{L-2} D(ABC, S) \quad (5.7)$$

kde L je opäť počet aminokyselín v proteíne, $D_{ABC,S}$ je deviácia tripeptidu ABC od bežných frekvencií rozpustnej sady. Prakticky to znamená, že autori mali určitú sadu trénovacích proteínov, ktoré boli rozpustné. Deviácia tripeptidu ABC od bežných frekvencií tejto sady sa počíta nasledovne:

$$D_{ABC,S} = log\left(\frac{F_{ABC,S}}{F_{ABC,nat}}\right) \quad (5.8)$$

kde $F_{ABC,S}$ je frekvencia výskytu tripeptidu ABC v rozpustnej dátovej sade (počítaná ako počet výskytov tripeptidov ABC vo všetkých vzorkoch danej sady, podelený počtom všetkých tripeptidov vo všetkých vzorkoch danej sady) a $F_{ABC,nat}$ je frekvencia výskytu tripeptidu ABC v proteínoch všeobecne (autori ju počítali na proteínoch z databázy *Swiss-Prot*).

Trénovanie a testovanie

Autori si na trénovanie vytvorili dve dátové sady, jednu obsahujúcu 25 rozpustných proteínov, druhú obsahujúcu 105 nerozpustných proteínov. Na testovanie vytvorili dodatočnú sadu s 15 rozpustnými a 25 nerozpustnými proteínmi, ktorá je disjunktná s trénovacími

sadami. Pre získanie týchto proteínov využili autori databázu *PubMed*, konkrétny postup sme opísali v časti 4.1.1. Autori pre overenie presnosti vypočítali index rozpustnosti pre všetky prvky všetkých sád a dodatočne testovali ich model metódou *jack-knife* testovania a *bootstrap* metódou.

Záver

Za významný považujeme výsledok testovania pomocou testovacej sady. S touto sadou dosiahli autori presnosť 60% pre rozpustné proteíny a 64% pre nerozpustné. Trénovacie aj testovacie sady však boli pomerne malé, neobsahujúc spolu ani 200 vzorkov. Vzhľadom k neuvedeniu niektorých podstatných informácií ohľadom výpočtu indexu rozpustnosti sme sa rozhodli tento model nevyužiť vo výslednom metaprediktore.

5.1.3 Idicula-Thomas et al. 2006

Princíp predikcie

Autori ([19]) pracovali s princípom *SVM* a vyskúšali niekoľko rôznych vstupných vektorov rysov. V prvom rade zostavili tri rôzne abecedy aminokyselín:

- **podobnosť priestorového zloženia** (*conformational similarity*): [CMQLEKRA], [P], [ND], [G], [HWFY], [S], [TIV],
- **hydrofóbnosť**: [CFILMVW], [AG], [PH], [EDRK], [NQSTY],
- **substitučná matica BLOSUM50**: [FWY], [CILMV], [H], [AG], [ST], [EDNQ], [KR], [P].

Tým získali spolu 20 počítateľných frekvencií výskytov aminokyselín, teda 20 rysov. Ďalších 6 rysov predstavovali fyzikálne-chemické vlastnosti proteínov:

- **L**: dĺžka proteínu,
- **GRAVY**: *hydropathic index*,
- **AI**: alifatický index,
- $\mathbf{II_P}$: index nestability,
- $\mathbf{II_N}$: index nestability N-konca,
- **NC**: celkový náboj.

K týmto 26 rysom autori pridali ešte frekvencie mono- a dipeptidov, čím dosiahli počet 446 rysov. Hodnoty všetkých rysov boli upravené tak, aby mali nulový priemer a odchýlku 1.

Autori vyskúšali model, kde použili len 46 rysov, vynechávajúc frekvencie dipeptidov, ako aj model s 446 rysmi a model s 8446 rysmi, kde k vyššie uvedeným 446 rysom pridali ešte frekvencie tripeptidov. Najlepšie výsledky priniesol práve model so 446 vstupnými rysmi, preto autori ďalej pracovali s týmto modelom.

Popri metóde *SVM* vyskúšali autori aj lineárnu logistickú regresiu a metódu K najbližších susedov, *SVM* metóda však priniesla najlepšie výsledky. Autori dodatočne vyskúšali aj váhované verzie *SVM* a metódy K najbližších susedov, pretože ich trénovacia sada bola

nevyvážená. Tieto pokusy priniesli mierne zlepšenie, ale autori ho nepovažovali za príliš podstatné a vzhľadom na vyššiu výpočetnú náročnosť váhovaných verzií pracovali ďalej s pôvodným modelom *SVM*.

V snahe zredukovať počet vstupných rysov využili autori metódu výpočtu *nevyváženého korelačného skóre* (*unbalanced correlation score*), ktorá spočíva vo výpočte prínosu konkrétneho rysu do celkovej predikcie. Následne autori vybrali 20 najdôležitejších rysov, korelujúcich s predikovanou rozpustnosťou (F(x) značí frekvenciu výskytu mono- alebo dipeptidu x): alifatický index, F(Glu), F(His-His), F(Arg-Gly), F(Arg), F(Gly), index nestability celého proteínu, celkový náboj, F(Asn-Thr), F(Arg-Ala), F(Cys), F(Met), F(Gln), F(Phe), F(Ile), F(Gly-Ala), index nestability N-konca, F(Ser), F(Leu) a F(Pro). Model *SVM* využívajúci týchto 20 rysov dosiahol približne rovnakú presnosť, ako model so 446 rysmi.

Trénovanie a testovanie

Autori podobne ako vo svojej predchádzajúcej práci získavali svoje dáta na základe vyhľadávania v literatúre. Získali takto 62 rozpustných a 130 nerozpustných proteínov, ktoré rozdelili do trénovej sady (41 rozpustných a 87 nerozpustných) a testovacej sady (21 rozpustných a 43 nerozpustných proteínov). Autori pre všetky dáta získali požadované rysy, upravili ich hodnoty tak, aby ich priemer bol nulový a štandardná odchýlka 1. Následne spustili *SVM* na trénovacej sade a potom overili presnosť na testovacej sade.

Záver

Autori uviedli pre svoje modely nasledujúce presnosti:

- 446 rysov: 72%,
- 46 rysov: 66%,
- 8446 rysov: 67%,
- 20 rysov: 70%.

Prediktor je dostupný na vyžiadanie, a preto sme požiadali autorov o zaslanie potrebných súčastí. Autori nám vyhoveli, zaslaný kód sa nám však žiaľ nepodarilo spustiť, nakoľko obsahoval chyby, ktoré sme neboli schopní odhaliť a opraviť. Vo výsledku sme teda tento prediktor nevyužili pri tvorbe metaprediktoru.

5.1.4 PROSO

Princíp predikcie

Autori ([32]) použili dvojfázovú predikciu, kde v prvej fáze využili *SVM* klasifikátory (o ktorých sa zmieňujeme v časti 3.3.5) s Gaussovským jadrom, ktorých výsledky následne figurujú ako vstup druhej fázy, kde je použitý klasifikátor na princípe *naivnej bayesovskej klasifikácie* (3.3.1). Výsledok druhej fázy už je konečným výsledkom klasifikátoru.

V prvej fáze podstúpili vstupné proteíny klasifikáciu pomocou troch rôznych *SVM* klasifikátorov, pričom každý pracoval s inými vstupnými rysmi proteínov. Jeden klasifikátor pracoval s frekvenciami jednotlivých aminokyselín, druhý s frekvenciami dvojíc a tretí s frekvenciami trojíc. Pre dvojice by však pri použití celej 20-znakovej abecedy vzniklo 400

rôznych rysov a pre trojice až 8000, preto autori namiesto celej abecedy pozhlukovali aminokyseliny s podobnými vlastnosťami do zhlukov a ďalej pracovali s frekvenciami týchto zhlukov. Vznikli dve zhlukovacie schémy:

- **sol14**: [S,T], [G], [R], [F,W], [M], [D,Q,E], [K], [Y], [P], [I,V], [L], [N], [H,A], [C],
- **sol17**: [S], [H], [T], [L,I], [W], [M], [F], [D,E], [A], [C], [K], [G], [P], [Y], [N,Q], [R], [A].

Navyše autori testovaním zistili, ktoré rysy najlepšie rozlišujú rozpustné a nerozpustné proteíny a vo výsledku teda nepoužívali všetky možné kombinácie.

Trénovanie a testovanie

Dátovú sadu vytvorili autori s využitím databáz *TargetDB* (4.1.3) a *PDB* (4.1.5), ako aj s využitím literárnych prameňov. Z databázy *TargetDB* vybrali tie proteíny, ktoré dosiahli aspoň stavu *soluble* a pridali ich medzi rozpustné proteíny, zatiaľ čo proteíny, ktoré v období od apríla 2005 do novembra 2005 zotrvali v stave *expressed*, zaradili do sady nerozpustných proteínov. Dodatočne vylúčili zo sady nerozpustných proteínov tie, pri ktorých našli 100% zhodu s nejakým záznamom v databázi *PDB*.

Všetky získané dáta následne preskúmali nástrojom *TMHMM*, o ktorom sa zmieňujeme v časti 4.2.1, a vylúčili tie proteíny, pri ktorých bola predikovaná prítomnosť transmembránových segmentov, ako aj tie, ktoré obsahovali viac než jednu neznámu aminokyselinu na viacerých susediacich pozíciach. Na záver autori odstránili redundanciu s využitím zhlukovacieho nástroja *CD-HIT* použitím 50% hranice podobnosti.

Výslednú dátovú sadu rozdelili na dve časti, kde jedna obsahovala len proteíny, obsahujúce jedinú doménu a druhá proteíny, obsahujúce viac domén. Do týchto skupín autori dáta rozdelili na základe dĺžky aminokyselinovej sekvencie a to konkrétne s hranicou 250 aminokyselín. Proteíny s dlhšími sekvenciami zaradili medzi mnohodoménové.

Nad prediktormi prvej úrovne a dátovou sadou bola vykonaná 10-násobná krížová validácia, a to osobitne na jednodoménovej a mnohodoménovej časti. Tento postup vyprodukoval výsledky prediktorov prvej úrovne pre všetky dáta, a tieto výsledky boli použité ako vstupy sekundárneho klasifikátoru. Autori nad vstupnými dátami vykonali 10-násobnú stratifikovanú krížovú validáciu a tak získali klasifikáciu pre každý proteín, ako aj odhad presnosti sekundárneho prediktoru.

Zhrnutie

Autori prediktoru ohlásili presnosť 71,7%, avšak autori prediktoru *SOLpro* (5.1.5) pri testovaní nad svojou dátovou sadou ukázali presnosť prediktoru *PROSO* len 59,28%. Nástroj je dostupný len on-line[2], čo komplikuje možnosť predikovať pre veľkú dátovú sadu a teda obmedzuje možnosti nášho využitia tohto prediktoru. Dôležitý je aj fakt, že prediktor má aj svojho nástupcu menom *PROSO II*, ktorému sa venujeme v časti 5.1.6.

5.1.5 SOLpro

Princíp predikcie

Predikciu vykonáva v prvom rade 20 *SVM* klasifikátorov, každý nad jednou sadou vybraných rysov. Výsledky prvej fázy predikcie spolu s dĺžkou aminokyselinovej sekvencie

[2] http://mips.helmholtz-muenchen.de/proso/proso.seam

predstavujú 21 vstupov druhej fázy predikcie, ktorú tvorí ďalší *SVM* klasifikátor. Autori ([24]) použili 23 sád rysov, z čoho 21 sú frekvencie mono-, di- a trimérov. Tieto frekvencie sú vystavané nad 7 rôznymi množinami aminokyselín, z čoho jedna obsahuje všetkých 20 aminokyselín a zvyšné sú redukované (napríklad množina hydrofóbnych aminokyselín). Zvyšné dve sady rysov majú názov *Computed* a *Predicted*. Vo výsledku ponechali autori len 20 z týchto 23 sád.

Sada *Computed* obsahuje rysy spočítateľné priamo zo sekvencie, a to dĺžku sekvencie, podiel aminokyselín tvoriacich otočky[3], absolútny náboj na aminokyselinu, molekulárnu hmotnosť, *GRAVY* (grand average of hydropathicity[4]) index a alifatický index. Sada *Predicted* obsahuje rysy, ktoré boli predikované zo sekvencie s využitím sady prediktorov *SCRATCH*[5]. Konkrétne ide o podiel beta listov, podiel alfa špirál (oboje predikované nástrojom *SSpro*), počet domén (predikované nástrojom *DOMpro*) a podiel aminokyselín, vystavených na povrchu proteínu (predikované nástrojom *ACCpro*).

Trénovanie a testovanie

Autori vytvorili dátovú sadu *SOLP*, ktorú detailne opisujeme v časti 4.2.1. Pre odhad presnosti prediktoru vykonali autori opakované 10-násobné krížové validácie pre rôzne náhodné rozdelenia ich dátovej sady. Rozdelenia boli vždy vyvážené, teda každá časť dátovej sady vždy obsahovala vyrovnaný počet rozpustných a nerozpustných proteínov.

Zhrnutie

Pre 10 behov 10-násobnej krížovej validácie ohlásili autori presnosť 74,15% a MCC 0,487. Pri našom testovaní na dátovej sade *SOLP* dosiahol podobnú presnosť (72,7%), pričom rozdiel môže byť spôsobený rozličnými verziami podporných programov *SSpro* a *DOMpro*, alebo inou verziou NR databázy, ktorú je nutné programu dodať. Prekvapivejšie sú výsledky testovania na našej dátovej sade pri podmienkach, ktoré sme opísali v úvode tejto sekcie. Pri tejto dátovej sade dosiahol prediktor presnosť len 53,91%, pričom počet testovacích vzorkov bol dvakrát 14 266.

5.1.6 PROSO II

Princíp predikcie

Podobne ako v prípade predchodcu tohto prediktoru, *PROSO* (5.1.4), využíva prediktor *PROSO II* ([31]) dvojfázovú architektúru. Namiesto *SVM* klasifikátorov však autori použili model Parzenovho okna a logistickú regresiu.

Prvá vrstva obsahuje dva klasifikátory založené na porovnávaní s hranicou, pričom jeden klasifikátor na princípe logistickej regresie pracuje s frekvenciami mono- a dipeptidov a druhý, na princípe Parzenovho okna, počíta podobnosť testovanej sekvencie s inými sekvenciami. Výsledky týchto dvoch klasifikátorov slúžia ako vstup klasifikátoru druhej vrstvy, ktorý opäť pracuje na princípe logistickej regresie. Klasifikátor prvej vrstvy, založený na logistickej regresii, využíva frekvencie nasledujúcich mono- a dipeptidov:

- R, N, D, C, Q, E, G, H, I, K, M, F, P, S, T, W, Y, V,

[3] turn-forming residues
[4] hydropathicity - the relative hydrophobicity or hydrophilicity of a compound
[5] http://scratch.proteomics.ics.uci.edu/

- AK, CV, EG, GN, GH, HE, IH, IW, MR, MQ, PR, TS, WD.

Na výber konkrétnych mono- a dipeptidov, súvisiacich s rozpustnosťou, využili autori rovnaký postup ako v prípade prediktoru *PROSO* ([32]). Spolu s týmito informáciami použili autori aj globálne rysy celej sekvencie, menovite dĺžku, *pI* (izoelektrický bod), *GRAVY*, *AI* (alifatický index) a *FI* (fold index). Autori však neuviedli, ktorý klasifikátor tieto rysy využíva.

Čo sa týka klasifikátoru na princípe Parzenovho okna, tento počíta podobnosť testovaného proteínu voči trénovacím proteínom, pri ktorých je známa rozpustnosť. Podobnosť má v tomto prípade podobu *BLASTP*[6] skóre. Konkrétne sa počíta 6 rôznych skóre, na základe ktorých sa vypočítajú 2 hodnoty podobnosti – podobnosť s rozpustnými dátami a podobnosť s nerozpustnými dátami. Konkrétne sa spomenutých 6 skóre počíta nasledovne:

- **a:** podobnosť testovanej sekvencie voči sade rozpustných dát,
- **b:** podobnosť testovanej sekvencie voči sade nerozpustných dát,
- **c:** podobnosť prvku rozpustnej sady voči zvyšku rozpustnej sady. Toto skóre je spočítané pre každý prvok rozpustnej sady,
- **d:** podobnosť prvku nerozpustnej sady voči zvyšku nerozpustnej sady. Toto skóre je spočítané pre každý prvok nerozpustnej sady,
- **e:** podobnosť prvku rozpustnej sady voči nerozpustnej sade. Toto skóre je spočítané pre každý prvok rozpustnej sady,
- **f:** podobnosť prvku nerozpustnej sady voči rozpustnej sade. Toto skóre je spočítané pre každý prvok nerozpustnej sady.

Uvedené skóre sa následne použijú na výpočet dvoch podobností:

$$SS = \frac{1}{hn} \sum_{i=0}^{n} \frac{1}{\pi} \left(1 + \left(\frac{a - c_i}{h} \right)^2 + \left(\frac{b - e_i}{h} \right)^2 \right) \tag{5.9}$$

$$SI = \frac{1}{hm} \sum_{j=0}^{m} \frac{1}{\pi} \left(1 + \left(\frac{b - d_j}{h} \right)^2 + \left(\frac{a - f_j}{h} \right)^2 \right) \tag{5.10}$$

kde navyše m a n sú veľkosti rozpustnej a nerozpustnej sady a h je parameter jemnosti nastavený na hodnotu 0,65. Tieto dve podobnosti sú potom použité vo finálnej rovnici:

$$f = \frac{SS}{SS + SI} \tag{5.11}$$

Pokiaľ hodnota tejto rovnice presiahne hranicu 0,5, proteín je označený za rozpustný, inak za nerozpustný.

[6] http://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE=Proteins

Trénovanie a testovanie

Autori opakovane zdôrazňujú veľkosť trénovacej dátovej sady, ktorú tvorí 82 299 proteínov s experimentálne zistenou rozpustnosťou. Túto dátovú sadu sme opísali v časti 4.2.2. Pri testovaní testovali autori ako samostatné klasifikátory prvej úrovne, tak aj celkový klasifikátor, a to na dátach zhlukovaných na 90% identitu, pričom tieto dáta boli vyvážené. Testovanie nad trénovacou sadou prebiehalo pomocou 10-násobnej stratifikovanej krížovej validácie. Dodatočne autori využili aj separátnu testovaciu sadu, ktorá nebola použitá pre trénovanie. Táto separátna dátová sada bola nevyvážená a obsahovala jednu rozpustnú sekvenciu na každých 5 nerozpustných. Autori týmto chceli simulovať reálnu distribúciu sekvencií.

Zhrnutie

Prediktor je dostupný len on-line[7]. Autori testovali presnosť jednotlivých klasifikátorov, ako aj celkovú presnosť celého klasifikátoru. Klasifikátor na základe frekvencií monopeptidov dosiahol presnosť 59,7%, na základe frekvencií dipeptidov 68,6%, na princípe Parzenovho okna 64,3% a celý prediktor *PROSO II* dosiahol presnosť 71% a MCC 0,421. Pri dodatočnom testovaní prediktoru na separátnej sade dosiahol presnosť 75,4% a MCC 0,39.

Podobne ako pri nástroji *SOLpro* sme pri testovaní nástroja *PROSO II* na ich vlastnej dátovej sade dosiahli podobnú presnosť (71,6%), avšak pri testovaní na našej dátovej sade, tak ako sme opísali v úvode tejto sekcie, dosiahol prediktor presnosť len 49,98%. Konkrétne obsahovala testovacia sada dvakrát 5 164 prvkov.

5.1.7 Huang et al.

Princíp predikcie

Autori vo svojom článku ([17]) navrhujú nový princíp predikcie, zvaný *SCM* (*scoring card method*). Ich prístup využíva pre predikciu rozpustnosti len frekvencie dipeptidov, na základe ktorých zostaví tabuľku, z ktorej je následne možné odvodiť pravdepodobnosť rozpustnosti proteínu. Autori postupovali v 4 krokoch:

1. vytvorenie dátových sád na trénovanie a testovanie,
2. vytvorenie počiatočnej bodovej matice,
3. optimalizácia bodovej matice,
4. predikcia rozpustnosti proteínov.

V prvom kroku autori rozdelili svoju dátovú sadu na trénovaciu a testovaciu, pričom trénovaciu použijú na tvorbu a optimalizáciu bodovej matice. Samotná tvorba počiatočnej matice prebieha taktiež v 4 krokoch:

1. spočítanie počtu každého zo 400 dipeptidov v oboch triedach. Výsledkom je teda 800 čísel – 400 pre frekvencie dipeptidov medzi rozpustnými sekvenciami a 400 pre nerozpustné sekvencie,

[7] http://mips.helmholtz-muenchen.de/prosoII/prosoII.seam

2. normalizácia počtov z prvého kroku. Tento krok spočíva v podelení každého získaného čísla celkovým počtom výskytov všetkých dipeptidov v danej triede. Číslo potom prakticky predstavuje podiel daného dipeptidu voči všetkým dipeptidom v danej triede a je obmedzené na rozsah 0 až 1,

3. odčítanie skóre dipeptidu pre nerozpustnú triedu od rozpustnej. Týmto krokom autori získajú opäť 400 čísel, pričom tieto čísla už predstavujú hodnoty bodovej matice,

4. normalizácia všetkých 400 skóre do rozsahu 0 až 1000.

Z tejto matice sa dá spočítať aj bodová matica jednotlivých aminokyselín. Napríklad pre aminokyselinu A (alanín) zoberieme skóre dipeptidov AX a dipeptidov XA, kde X je ľubovoľná aminokyselina, a tieto hodnoty spriemerujeme. Po vytvorení počiatočnej bodovej matice je nutné ju optimalizovať. To autori spravili pomocou evolučného algoritmu *IGA*.

Predikcia samotná prebieha tak, že sa spočíta podiel každého dipeptidu voči celkovému počtu dipeptidov, čím získame 400 hodnôt v rozsahu od 0 do 1, ktoré si označme w_i pre i od 1 do 400. S_i pre i od 1 do 400 predstavuje hodnoty z bodovej matice pre dipeptidy. Skóre rozpustnosti sa potom počíta nasledovne:

$$S = \sum_{i=1}^{400} w_i S_i \tag{5.12}$$

Pokiaľ hodnota S prekročí stanovený prah, považujú autori daný proteín za rozpustný, inak za nerozpustný.

Trénovanie a testovanie

Autori v práci spomínajú 4 dátové sady, z čoho jedna je podmnožinou inej, takže prakticky pracujú autori s tromi rôznymi sadami. Prvou je sada, nazvaná *Sd95%*, ktorá obsahuje len proteíny, u ktorých bola rozpustnosť potvrdená biologickými experimentmi. Tieto proteíny získali autori z literatúry, a to ako priamo z databázy *PubMed*, tak aj z iných prác zaoberajúcich sa rozpustnosťou proteínov, menovite [8] a [6]. Zvyšné dve dátové sady tvoria sady *SOLP* a *PROSO II*, ktoré sme spomínali v častiach 4.2.1 a 4.2.2. Primárne využívali autori sadu *Sd95%*, ktorú rozdelili náhodne na 766 trénovacích a 191 testovacích prvkov. Dodatočne testovali autori prediktor aj nad sadami *SOLP* a *PROSO II*.

Záver

Priemerná hodnota 10-násobnej krížovej validácie nad trénovacou množinou dosiahla presnosť 83,37%, priemerná hodnota pri testovacej sade bola 83,98%. Pri testovaní nad sadou *SOLP* hlásia autori presnosť 59,99%, pri použití sady *PROSO II* dosiahli presnosť 64,5%.

Metóda poskytuje nový pohľad na predikciu rozpustnosti a autori uvádzajú zaujímavé presnosti. Autori taktiež zverejnili svoj prediktor, uvedená stránka však už žiaľ neexistuje a nepodarilo sa nám tieto zdrojové kódy na internete dohľadať. Preto sme tento prediktor pri tvorbe metaprediktoru nevyužili.

5.1.8 ccSOL

Princíp predikcie

Pri tvorbe prediktoru *ccSOL* ([2]) využili autori metódu *SVM*. Pre každú sekvenciu ich dátovej sady spočítali 28 fyzikálne-chemických vlastností, súvisiacich s alfa špirálami, beta

listami, beta otočkami, neusporiadanosťou, vinutím sekvencie, hydrofóbnosťou a inými oblasťami. Následne rozdelili autori svoju dátovú sadu na tri časti: veľmi rozpustné proteíny, málo rozpustné proteíny a zvyšok.

Autori potom vytvorili *SVM*, ktorý dostal za úlohu rozlíšiť rysy, ktoré dobre rozdeľujú veľmi rozpustné a málo rozpustné proteíny. Iteratívnym procesom dospeli autori k 11 rysom, ktoré dobre rozlišovali rozpustné od nerozpustných proteínov. Medzi najvýznamnejšie rysy patrí neusporiadanosť proteínu, jeho vinutie a jeho hydrofília. Po získaní tejto menšej množiny rysov vytvorili autori 2^{11} rôznych *SVM*, teda jeden pre každú kombináciu 11 rysov. Všetky tieto modely otestovali metódou 10-násobnej krížovej validácie a zistili tak, ktorý model poskytuje najvyššiu presnosť. Najlepšie výsledky dosiahol model so 6 rysmi: vinutím proteínu, neusporiadanosťou proteínu, jeho hydrofóbnosťou, hydrofíliou, beta otočkami a alfa špirálami.

Trénovanie a testovanie

Autori práce využili dátovú sadu od autorov Niwa et al. (4.2.3). Táto sada obsahovala 3043 proteínov, ktoré autori rozdelili na 3 sady, ako sme uviedli v predchádzajúcej časti. Vytvorených 2048 *SVM* modelov testovali autori metódou 10-násobnej krížovej validácie.

Záver

Autori neuviedli namerané výsledky testovania, ale poskytli graf, z ktorého sme vyčítali hodnotu približne 93% pre model so šiestimi rysmi. Neskorší článok autorov ([1]) uvádza pre prediktor *ccSOL* presnosť 76%. Metóda je dostupná len on-line[8] a pre naše účely trvá výpočet príliš dlho, preto sme tento prediktor v našej práci ďalej nevyužili.

5.1.9 Fang et al.

Princíp predikcie

Na predikciu rozpustnosti využili autori ([9]) metódu *náhodných lesov* (*random forest*), ktorá využíva mnoho nezávislých rozhodovacích stromov, z ktorých potom metóda vytvára jediný výsledok. Autori použili 5000 rozhodovacích stromov, pričom každý z nich trénovali na osobitnej podmnožine celej trénovacej sady, pričom každá táto podmnožina mala veľkosť $\sqrt{M}$, kde M je veľkosť celej trénovacej sady. Prvky do týchto podmnožín vyberali autori metódou *bootstrap*, takže tieto podmnožiny neboli disjunktné a mohli obsahovať niektoré prvky aj viackrát v rámci jednej podmnožiny.

Každý proteín bol reprezentovaný sadou 1438 rysov, ktoré spadajú do 4 kategórií:

1. fyzikálne-chemické vlastnosti,
2. absolútne a normalizované počty aminokyselín,
3. absolútne a normalizované počty dipeptidov,
4. zvyšné rysy.

Všetky rysy sú počítané z primárnej sekvencie aminokyselín, prípadne zo štruktúrnych informácií, získaných predikciou z primárnych sekvencií. Pre zhodnotenie významnosti jednotlivých skupín rysov vyskúšali autori vytvoriť modely s rôznymi kombináciami uvedených

[8] http://s.tartaglialab.com/new_submission/ccsol

kategórií rysov. Najkvalitnejšie výsledky dosiahol model, ktorý využíval všetky 4 kategórie, čo naznačuje, že všetky skúmané rysy nejakou mierou súvisia s rozpustnosťou proteínov, pričom aminokyselinové zloženie je pre predikciu najdôležitejšie a dipeptidové zloženie zas najmenej dôležité. Autori sa následne snažili zminimalizovať počet vstupných rysov, na čo využili balíček *varSelRF* jazyka *R*. Týmto postupom získali 17 najvýznamnejších rysov, ktoré súvisia s rozpustnosťou proteínov.

Trénovanie a testovanie

Vo svojej práci použili autori dáta z databázy *eSOL*, ktorú spomíname v časti 4.1.2. Táto dátová sada obsahuje namiesto binárnej hodnoty rozpustnosti percentuálnu hodnotu. Autori ju previedli na binárnu tak, že proteíny s rozpustnosťou nižšou než 30% označili za nerozpustné, zatiaľ čo proteíny s rozpustnosťou nad 70% považujú za rozpustné. Po tejto úprave zostalo 2 183 proteínov, táto sada sa ďalej zmenšila po zhlukovaní na úroveň 30% identity na 1918 proteínov. Autori žiaľ neuviedli spôsob, akým prediktor testovali.

Záver

Autori ohlásili presnosť predikcie 84%. Prediktor bol zverejnený na internete, bohužiaľ je však uvedený odkaz neplatný a podobne ako v prípade prediktoru od autorov Huang et al. (5.1.7) sa nám nepodarilo tento prediktor nájsť inde na internete, takže sme ho nevyužili pri tvorbe metaprediktoru.

5.1.10 ESPRESSO

Princíp predikcie

Prediktor ([15]) pracuje s dvomi metódami predikcie:

- predikcia na základe vlastností sekvencie a predikovanej štruktúry,
- predikcia na základe vzorov v sekvencii.

Čo sa týka predikcie na základe vlastností sekvencie a predikovanej štruktúry, autori najprv definovali 437 rysov na základe dvoch typov informácií – jednak na základe samotnej sekvencie (napr. nukleotidové zloženie, aminokyselinové skupiny, atď.) a jednak na základe štruktúry, predikovanej rôznymi nástrojmi (*PHD*[9], *RVP-net*[10] a iné).

Pre každý rys autori ďalej spočítali štatisticky významnú odchýlku medzi pozitívnymi a negatívnymi dátovými vzorkami pomocou Studentovho t-testu a rysy s $p < 0.05$ považovali za súvisiace s expresiou a rozpustnosťou proteínov. Vo výsledku zostalo 63 rysov súvisiacich s expresiou a 50 rysov súvisiacich s rozpustnosťou proteínov v E. coli. Boli vyskúšané tri rôzne prístupy strojového učenia – SVM, náhodné lesy a neurónové siete – a z nich autori vybrali najlepší model, ktorým sa ukázal byť SVM.

Druhá metóda je predikcia na základe vzorov. Táto metóda využíva frekvencie výskytov často sa vyskytujúcich vzorov v sekvenciách. Autori našli sekvenčné vzory, ktoré sa objavovali len v pozitívnych alebo len v negatívnych vzorkách. Po preskúmaní vzťahu medzi sekvenčnými vzormi a presnosťou predikcie autori zistili, že optimálna veľkosť sady pozitívnych vzorov (teda tých, ktoré súvisia s dobrou rozpustnosťou) je 734 vzorov, pre negatívne vzory je to 149.

[9] https://npsa-prabi.ibcp.fr/cgi-bin/npsa_automat.pl?page=/NPSA/npsa_phd.html
[10] http://www.abren.net/netasa/rvp-net-2/

Trénovanie a testovanie

Autori využili dátovú sadu z ich predchádzajúcej práce ([14]), ktorá bola vytvorená na základe experimentov, ktoré zisťovali expresiu a rozpustnosť ľudských cDNA fragmentov v E. coli a iných expresných systémoch. Dátová sada bola rozdelená na dve časti na základe počtu experimentov s konkrétnou sekvenciou. Jednu časť tvorili sekvencie, na ktorých bola expresia a rozpustnosť zistená len jedným experimentom (*dataset_S*). Druhú časť potom tvorili sekvencie, pri ktorých bola expresia a rozpustnosť zisťovaná dvomi alebo viacerými experimentami (*dataset_M*). Následne bola na oboch skupinách odstránená redundancia pomocou zhlukovania s využitím programu *CD-HIT* na 80% a následne 40% identitu. Dodatočne bola redundancia znížená ešte na základe globálneho zarovnania a to na hranici 25% identity. Vo výsledku obsahovala dátová sada 5 100 proteínov, dátové sady však neboli vyrovnané (obsahovali približne dvakrát viac nerozpustných než rozpustných sekvencií).

Metóda založená na vlastnostiach sekvencie a na predikovanej štruktúre bola testovaná jednak pomocou 10-násobnej krížovej validácie s využitím dátovej sady *dataset_M* a jednak pomocou dátovej sady *dataset_S*. V druhom prípade už nebola použitá krížová validácia, pretože metóda bola trénovaná na dátovej sade *dataset_M*, ktorá nemá prekryv s dátovou sadou *dataset_S*. Metóda založená na vzoroch bola testovaná na dátovej sade *dataset_M*, opäť bez krížovej validácie, keďže táto metóda bola trénovaná na separátnej dátovej sade *dataset_S*.

Zhrnutie

Autori ohlásili presnosť predikcie rozpustnosti 68% a MCC 0,42 pri metóde založenej na vlastnostiach sekvencie a 63% (MCC 0,23) pri metóde založenej na vzoroch. Pri testovaní na našej dátovej sade sú žiaľ výsledky horšie: 49,33% pre metódu založenú na vlastnostiach a 51,78% pri metóde založenej na vzoroch. Nástroj je dostupný on-line[11] a autori poskytujú aj skript v jazyku Perl pre automatizované spracovávanie sekvencií.

5.1.11 Zhrnutie

Ako sme spomenuli už v úvode tejto časti, mnoho prediktorov nebolo vôbec dostupných, či už z dôvodu, že vznikli len pre účely daného výskumu, alebo preto, že odkazy na tieto prediktory sú neplatné. Z uvedených prediktorov sa ukazuje, že model *SVM* je v oblasti predikcie rozpustnosti veľmi populárny a bol použitý aj vo viacerých prediktoroch, ktoré sme tu neuviedli. Medzi časté zdroje dát patrí *TargetTrack* alebo niektorý z jeho predchodcov a prekvapivo veľa autorov čerpalo dáta z literatúry.

Tabuľka 5.1 zhŕňa informácie o prediktoroch spomenutých v predchádzajúcom texte. Pre všetky prediktory s výnimkou *Davis et al.* uvádzame presnosti, ktoré uviedli autori vo svojich článkoch. Niektorí autori uviedli viaceré presnosti, pokiaľ napríklad testovali svoj prediktor na viacerých rozličných dátach, prípadne pokiaľ testovali viaceré metódy predikcie. Pri prediktoroch, ktoré sme sami použili, uvádzame aj nami namerané presnosti.

5.2 Použité prediktory

Z vyššie uvedených prediktorov sme využili štyri: Davis et al., SOLpro, PROSO II a ESPRESSO, z čoho dva sú on-line (PROSO II a ESPRESSO) a dva off-line.

[11] http://cblab.meiyaku.jp/ESPRESSO/Submission.php

Názov	Dátová sada	Model strojového učenia	Presnosť (uvedená autormi/nameraná nami)
Davis et al.	-	Regresia	-/52%
Idicula-Thomas et al. 2005	literatúra	Heuristická rovnica	62%/-
Idicula-Thomas et al. 2006	literatúra	SVM	70%/-
PROSO	TargetDB, PDB, literatúra	SVM v dvoch fázach	71,7%/-
SOLpro	PDB, SwissProt, TargetDB, literatúra	SVM v dvoch fázach	74,15%/53,91%
PROSO II	pepcDB	Parzenovo okno a regresia	75,4%/49,98%
Huang et al.	literatúra, SOLP, PROSO II	SCM	83,98%, 59,99%, 64,5%/-
ccSOL	Niwa et al.	SVM	93%/-
Fang et al.	eSOL	Náhodný les	84%/-
ESPRESSO	vlastné experimenty	SVM	68%, 63%/49,33%, 51,78%

Tabuľka 5.1: Prehľad prediktorov rozpustnosti.

- **Davis et al.:** Prediktor sme implementovali v jazyku C++ a ide v podstate o program, ktorý spracováva vstupný súbor, počíta požadované rysy a vyhodnocuje rovnicu uvedenú v časti 5.1.1,
- **SOLpro:** Prediktor závisí na programoch *SSpro 4.0*[12] a *DOMpro 1.0*, ako aj na knižnici *LibSVM*[13]. Pre správnu funkčnosť je nutné dodať aj NR (non-redundant) databázu, dostupnú napríklad na FTP serveri *NCBI*[14]. Program sme nastavili a používali v systéme Ubuntu 15.10. Pri veľkých počtoch vstupných dát sme využili služby organizácie MetaCentrum,
- **PROSO II:** Tento prediktor je dostupný len on-line a neobsahuje žiadne rozhranie okrem grafického, ktoré by sa dalo využiť pre spracovanie veľkého množstva dát. Na podnet autora prediktoru sme využili nástroj na testovanie web-stránok, konkrétne *Selenium*[15] v podobe knižnice do jazyka Python. Pomocou tohto nástroja sme spustili prehliadač *Mozilla Firefox*, načítali stránku prediktoru a automatizovane vkladali sekvencie, následne ich odosielali na server a počkali na odpoveď. Túto sme analyzovali a získali z nej informácie o predikovanej rozpustnosti,

[12] http://download.igb.uci.edu/
[13] http://www.csie.ntu.edu.tw/~cjlin/libsvm/
[14] ftp://ftp.ncbi.nih.gov/blast/db/FASTA/
[15] http://www.seleniumhq.org/

- **ESPRESSO:** Ako sme spomenuli, prediktor je dostupný on-line a poskytuje grafické rozhranie, podobne ako nástroj *PROSO II*. Autori však poskytli aj skript v jazyku Perl, ktorý na vstupe očakáva súbor v autormi špecifikovanom formáte, ktorý spracuje, informácie potom posiela na server a prijíma od serveru výsledky, ktoré zapisuje do výstupného súboru.

Pre objektívne porovnanie týchto štyroch prediktorov sme z našej výslednej dátovej sady vybrali len tie záznamy, ku ktorým sme získali predikcie od všetkých štyroch prediktorov. Táto sada bola nevyvážená, a to v pomere 4 591 rozpustných ku 9 677 nerozpustným proteínom. Túto sadu sme teda vyvážili vyradením náhodných nerozpustných sekvencií tak, aby boli obe triedy zastúpené rovnako. Na výslednej sade s 9 182 prvkami sme teda dodatočne otestovali všetky štyri prediktory a výsledky ukazujeme v tabuľke 5.2.

Prediktor	Presnosť
Davis et al.	49,67%
SOLpro	49,34%
PROSO II	49,62%
ESPRESSO - property	46,10%
ESPRESSO - motif	45,68%

Tabuľka 5.2: Dodatočné testovanie prediktorov.

5.3 Dodatočná analýza dát

Pri testovaní prediktorov na našej dátovej sade sme získali slabé presnosti, ako vidno v tabuľke 5.1, pričom sa tieto namerané presnosti nezhodovali s presnosťami, ktoré uviedli autori niektorých prediktorov. Vykonali sme teda dodatočnú analýzu našich dát z pohľadu sekvenčnej podobnosti. Pri prediktoroch *SOLpro* a *PROSO II* zverejnili autori aj konkrétne sady, na ktorých prediktory trénovali a testovali, preto sme pri dátových sadách týchto dvoch prediktorov mohli vykonať dôkladnejšiu analýzu.

Ako sme už spomenuli, pri testovaní na našej dátovej sade dosiahli prediktory nízke presnosti (*PROSO II* 58% a *SOLpro* 56,17%). Je nutné poznamenať, že prediktory boli testované práve na dátach, ktoré boli disjunktné s dátami, ktoré pri tvorbe daných prediktorov využívali autori. Keď sme však naopak skúsili testovať prediktory *PROSO II* a *SOLpro* na dátach, ktoré použili autori pri danom konkrétnom prediktore, presnosť výrazne stúpla a dosiahla približne hodnoty, uvádzané autormi (*PROSO II* na dátovej sade autorov dosiahlo 71,6%, *SOLpro* dosiahlo 72,7%). Tieto výsledky naznačujú, že prediktory boli pretrénované na konkrétny typ dát a naša dátová sada obsahuje dáta s inými charakteristikami.

Pre účely porovnania dátových sád sme každú zo siedmych pôvodných sád osobitne zhlukovali na úroveň 40% identity. Následne sme z každej sady vybrali náhodných 2 000 prvkov ako reprezentantov danej sady. Zlúčením týchto siedmich množín sme získali sadu so 14 000 sekvenciami. Pre účely analýzy dátových sád *SOLpro* a *PROSO* sme pre každú sekvenciu tejto menšej dátovej sady spočítali všetky rysy, ktoré pre predikciu využívajú prediktory *PROSO II* a *SOLpro*. V prípade prediktoru *PROSO II* išlo o 31 rysov (frekvencie niektorých mono- a dimérov), v prípade prediktoru *SOLpro* išlo až o 368 rysov (frekvencie niektorých mono-, di- a trimérov pri použití 7 rôznych abecied, charakteristiky sekvencie,

ako napr. alifatický index, dĺžka alebo náboj a predikované vlastnosti sekvencie, ako pomer alfa špirál alebo pomer aminokyselín vystavených na povrchu proteínu).

Spočítali sme teda dané rysy pre všetkých 14 000 sekvencií. Pre každý rys sme potom pre jednotlivé prediktory vytvorili histogramy, porovnávajúce rozloženie daného rysu v dátovej sade autorov oproti rozloženiu v zvyšku dát. Pre rysy prediktoru *PROSO II* sme teda vykreslili v jednej časti histogramu spočítaný rys pre dáta z dátových sád *PROSO* a *PROSO ccSOL*, v druhej časti histogramu potom pre ostatných 5 dátových sád. Analogicky pre rysy prediktoru *SOLpro* sme oddelene vykreslili spočítané hodnoty pre dáta zo sád *SOLpro* a *SOLpro ccSOL* a oddelene pre zvyšných 5 sád. Vzniklo teda 31 histogramov pre rozdiel dátovej sady *PROSO* a zvyšku dát, a 368 histogramov pre rozdiel sady *SOLpro* od zvyšku dát.

Pre lepšiu kvantifikáciu výsledkov sme využili *chi-squared test* pre dve náhodné rozloženia dát. Narazili sme však na problém, kde pri tvorbe histogramov vznikalo priveľa košov bez prvkov. Toto bolo spôsobené tým, že dáta obsahovali niektoré hodnoty výrazne vzdialené od ostatných. Napríklad pri analýze frekvencií arginínu v dátovej sade *PROSO* sa medián dát pohyboval na hodnote 13, pričom prvý kvartil bol na hodnote 8 a tretí na hodnote 21. Táto sada však obsahovala aj prvky s hodnotou 102, ktoré teda výrazne vybočovali z distribúcie väčšiny dát, v dôsledku čoho vzniklo pri tvorbe histogramu mnoho prázdnych košov, s ktorými si algoritmus pre výpočet chi-squared testu nevedel poradiť.

Rozhodli sme sa teda pri výpočte chi-squared testu vyradiť z dát všetky odľahlé hodnoty (pre každý rys osobitne), čím sme výrazne znížili počet prázdnych košov. Za odľahlé hodnoty sme považovali tie, ktoré boli menšie než prvý kvartil o viac než 1,5-násobok medzikvartilovej vzdialenosti (teda rozdielu tretieho a prvého kvartilu) a podobne väčšie než tretí kvartil o viac než 1,5-násobok medzikvartilovej vzdialenosti.

Pri tvorbe histogramov sme využili pevný počet košov, a to 150. Pri výpočte chi-squared testu však bolo toto číslo priveľké a aj po odstránení odľahlých hodnôt vznikali prázdne koše, tentokrát hlavne z dôvodu, že konkrétna dátová sada bola vždy menšia, než zvyšok dát, a teda vznikali koše, ktoré pre väčšiu dátovú sadu obsahovali prvky, ale pre menšiu boli prázdne. Toto sme vyriešili adaptívnym znižovaním počtu košov tak, aby každý kôš obsahoval aspoň päť prvkov. Toto však bohužiaľ viedlo niekedy až do takých extrémov, kde z pôvodných 150 košov zostal jediný a v tomto prípade výpočet chi-squared testu nie je možné brať príliš vážne. Mnoho rysov však dosiahlo rozumný počet košov od 6 do 18 pri sade *SOLpro* a 6 až 37 pri sade *PROSO*.

Výsledky chi-squared testu ukázali, že v mnohých prípadoch je distribúcia daného rysu pre danú dátovú sadu rozdielna než pre zvyšok dát a podporujú teda názor, že daný prediktor je pretrénovaný na určitý typ dát. Tabuľka 5.3 ukazuje niekoľko rysov prediktoru *PROSO II* s najnižšími *p-hodnotami*, tabuľka 5.4 podobne pre prediktor *SOLpro*. Vybrali sme najnižšie hodnoty len z tých výsledkov, ktoré pracovali s aspoň 6 košmi.

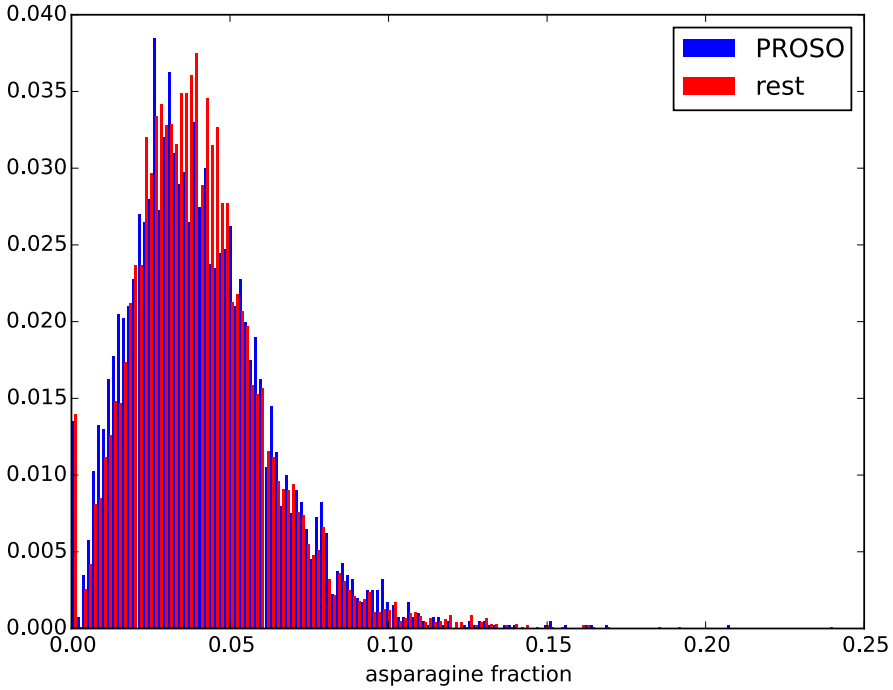
Rys	P-hodnota
Podiel asparagínu	$1,5153 \cdot 10^{-41}$
Podiel serínu	$1,2326 \cdot 10^{-40}$
Podiel kyseliny glutámovej	$1,0123 \cdot 10^{-39}$
Podiel tyrozínu	$1,9406 \cdot 10^{-39}$
Podiel izoleucínu	$1,0338 \cdot 10^{-23}$

Tabuľka 5.3: Výsledky chi-squared testu pre sadu *PROSO*.

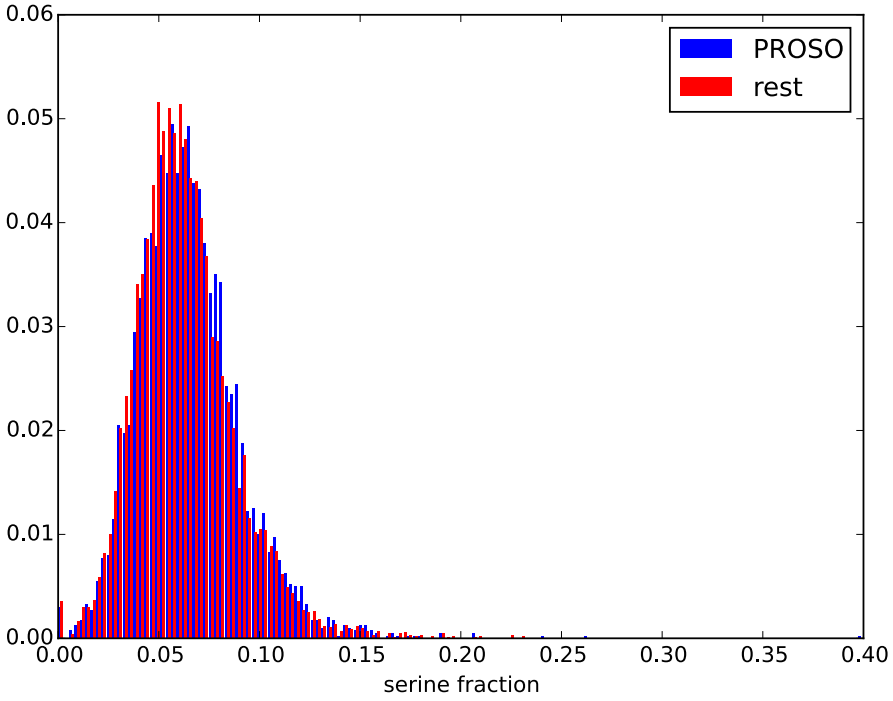
Rys	P-hodnota
Podiel monomérov kyseliny aspartánovej, kyseliny glutámovej, lyzínu a arginínu	$5,0945 \cdot 10^{-121}$
Podiel monomérov alanínu a histidínu	$8,895 \cdot 10^{-105}$
Podiel kyseliny glutámovej	$1,3452 \cdot 10^{-95}$
Podiel serínu	$1,7257 \cdot 10^{-84}$
Podiel monomérov kyseliny aspartánovej, kyseliny glutámovej a glutamínu	$4,6999 \cdot 10^{-84}$

Tabuľka 5.4: Výsledky chi-squared testu pre sadu *SOLpro*.

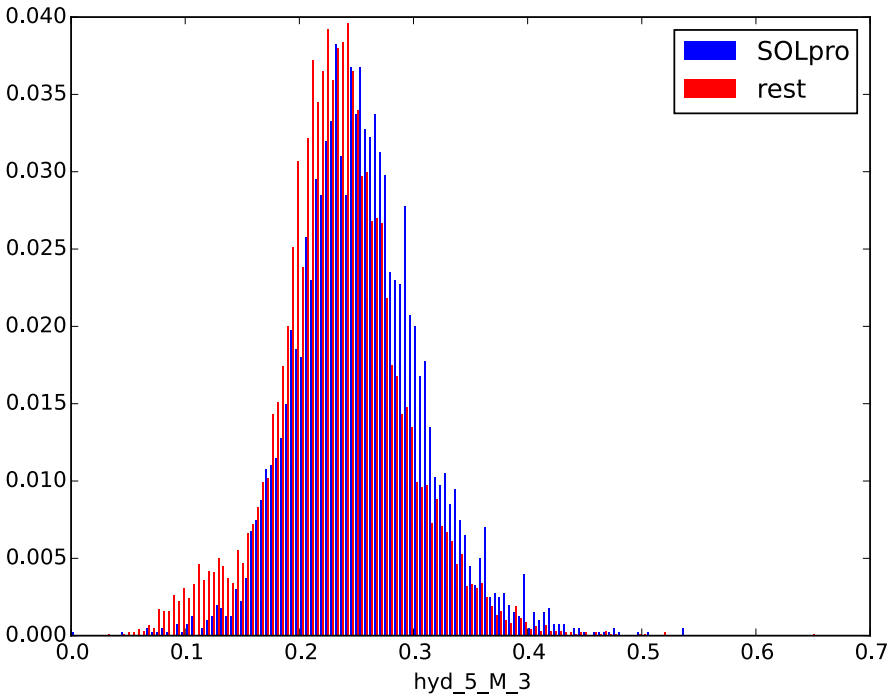
Celkovo 209 zo spomínaných 368 rysov získalo p-hodnotu menšiu než 0,05 pri sade *SOLpro*, pri sade *PROSO* to bolo 22 rysov z 31. Tieto počty síce zahŕňajú aj niektoré prípady, keď boli rysy rozdelené do malého počtu košov, ale napriek tomu tieto výsledky naznačujú rozdielnosť dátových sád použitých autormi od zvyškov sád. Pre ilustráciu ukazujeme dva histogramy pre každý z dvoch spomenutých prediktorov, ktoré ukazujú rysy s najnižšími p-hodnotami. Je nutné podotknúť, že tieto histogramy obsahujú aj odľahlé hodnoty, ktoré sme pri výpočte chi-squared testu vynechali. Výskyty sú taktiež normalizované, takže vertikálna osa zobrazuje pomer výskytu daného počtu oproti celkovým výskytom (teda napríklad hodnota 0,1 pri koši s hodnotou 5 by znamenala, že hodnota 5 predstavuje 10% zo všetkých dát danej sady).



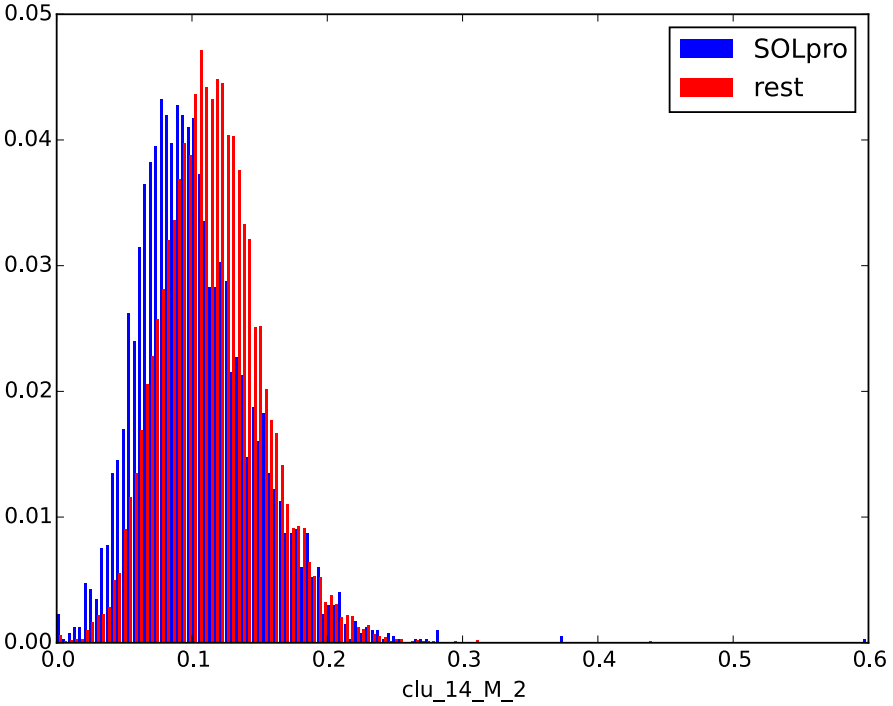
Obr. 5.1: Podiel asparagínu v sade *PROSO* a zvyšku dát



Obr. 5.2: Podiel serínu v sade *PROSO* a zvyšku dát



Obr. 5.3: Podiel monomérov kyseliny aspartánovej, kyseliny glutámovej, lyzínu a arginínu v sade *SOLpro* a zvyšku dát



Obr. 5.4: Podiel monomérov alanínu a histidínu v sade *SOLpro* a zvyšku dát

Kapitola 6

Implementácia

Celý proces tvorby metaprediktoru sprevádzali a umožňovali mnohé skripty, vykonávajúce rôzne úlohy od spracovania a analýzy pôvodných dátových sád, cez vytvorenie našej výslednej dátovej sady, testovanie prediktorov, výpočet rôznych metrík až po samotnú tvorbu výsledného konsenzuálneho metaprediktoru a jeho testovanie. V tejto kapitole opíšeme jednotlivé implementačné fázy a skripty, ktoré boli v týchto fázach použité. Všetky skripty boli napísané v jazyku *Python* a väčšina z nich predstavuje buď spracovanie textových súborov alebo prácu s databázou. Všetka implementačná práca bola vykonaná v prostredí operačného systému *Ubuntu 15.10*, ktorý bol spustený ako virtuálny stroj v programe *VirtualBox* spoločnosti *Oracle*. Celkový proces práce od úvodných fáz až po finálne testovania prehľadne ukazuje obrázok 6.1.

6.1 Príprava a analýza dát

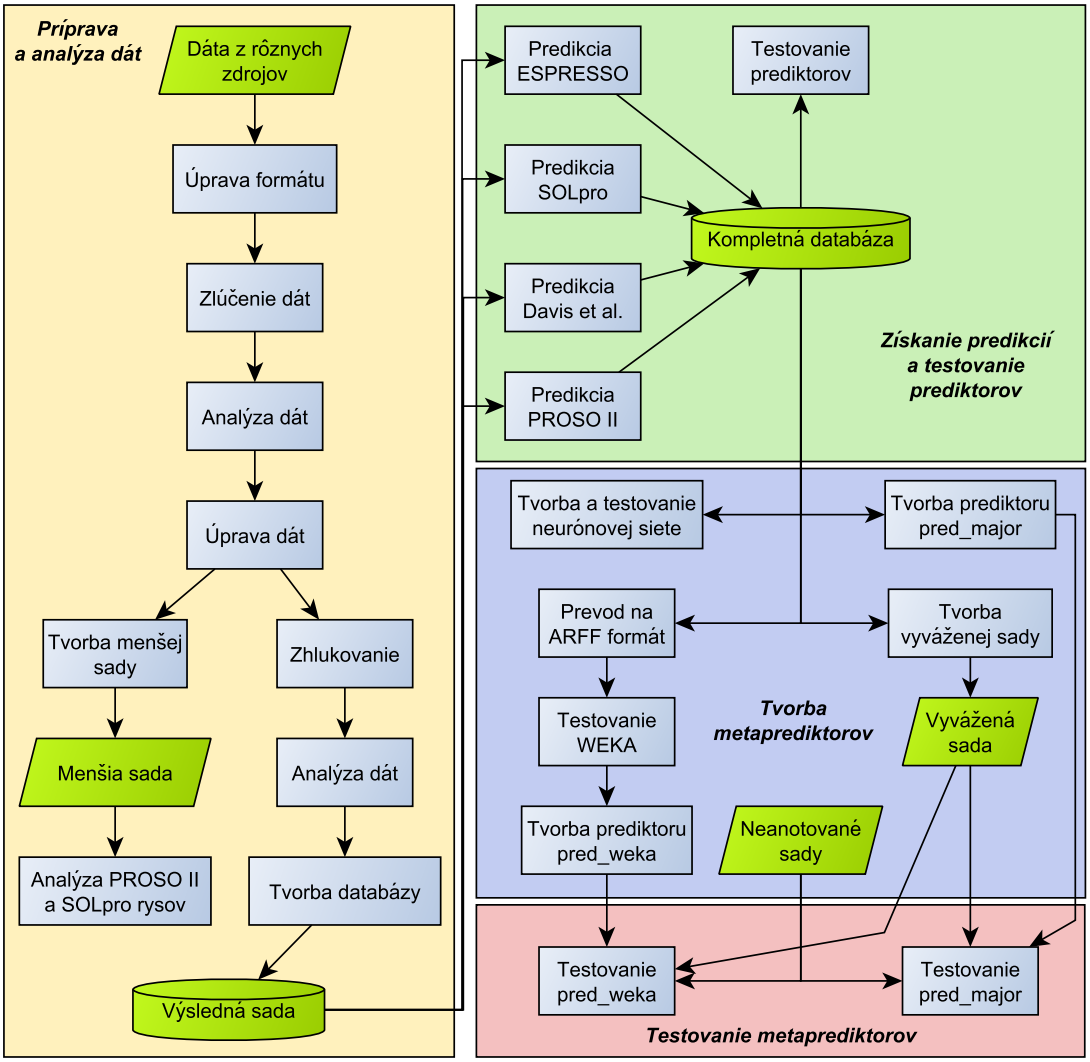
Ako sme napísali už v časti 4.3, väčšina pôvodných sád bola vo vhodnom tvare, s výnimkou sady *eSOL*, ktorú sme museli upraviť. Keď boli všetky dátové sady vo vhodnom tvare, zlúčili sme ich do jedinej veľkej dátovej sady. Výsledkom zlúčenia bol jediný súbor vo formáte FASTA, ktorého hlavičky mali nasledujúci formát:

```
>ID|JW|ECK|SP|SOLP|SOLB|TarDB|PDB|SIT|TT
```

kde `ID` predstavuje naše vlastné unikátne ID, ktoré začína od 1 a zvyšuje sa pre každú ďalšiu sekvenciu. Položky `JW`, `ECK`, `SP`, `TarDB`, `PDB`, `SIT` a `TT` predstavujú rôzne použité identifikátory pôvodných dátových sád, konkrétne *GenoBase ID*[1], *ECK prístupové čísla*, spomenuté v časti 4.3.1, *Swiss-Prot ID*, *TargetDB ID*, *PDB ID*, *ID dát od autorov Idicula-Thomas a Balaji* (tieto identifikátory boli použité autormi dátovej sady *SOLpro*) a *TargetTrack ID*. Každá dátová sada využívala iné identifikátory a niektoré sady nevyužívali žiadne identifikátory, preto jediným identifikátorom, ktorý sa nachádza pri každej sekvencii výslednej dátovej sady, je naše unikátne ID. Položky `SOLP` a `SOLB` predstavujú hodnoty rozpustnosti, a to buď v percentuálnej forme (túto položku majú len sekvencie z dátovej sady *eSOL*) alebo v binárnej forme (túto položku majú všetky sekvencie).

Z výslednej sady sme vyradili všetky sekvencie dlhšie než 2 000 aminokyselín. Ďalej sme napísali skripty, ktoré v tejto zlúčenej dátovej sade spočítali prekryvy dvojíc dátových sád a ktoré počítajú kolízie v dátach, kde rôzne výskyty rovnakej sekvencie sú anotované

[1] http://ecoli.naist.jp/GB/



Obr. 6.1: Postup práce.

rôznymi hodnotami rozpustnosti. Po získaní týchto informácií sme kolíznym sekvenciám zmenili anotáciu a označili ich za rozpustné.

Ďalej sme pre účely analýzy vytvorili menšiu vzorku našej dátovej sady, ktorá obsahovala 14 000 sekvencií. Tvorbu tejto sady sme opísali v časti 4.3.4. Túto menšiu sadu sme využili pre analýzu podobnosti distribúcií jednotlivých rysov prediktorov *PROSO II* a *SOLpro* v dátových sadách, použitých týmito nástrojmi, a v ostatných dátových sadách. Tieto skripty využívali balíček `numpy` pre niektoré výpočty, ako aj balíček `matplotlib` a konkrétne jeho súčasť `pyplot`, ktorý sme využili pri tvorbe grafov. K tejto analýze sme okrem grafickej podoby využili aj výpočet chi-squared testu a v týchto skriptoch sme použili balíček `scipy` a jeho súčasť `chi2_contingency`.

Okrem práce s našou menšou dátovou sadou sme vykonali aj analýzu pôvodných dátových sád na celých pôvodných dátových sadách. Tu sme pre každú pôvodnú sadu spočítali vzájomné podobnosti dvojíc. Na to sme využili program `usearch` a to nasledovným

spôsobom:

```
$ ./usearch -makeudb_usearch IN.fasta -output DB.udb
$ ./usearch -usearch_global IN.fasta -db DB.udb -id 0 -self -userout OUT.out
-userfields query+target+id -maxaccepts 10
```

kde IN.fasta predstavuje vstupný súbor vo fasta formáte. Prvý riadok vytvorí databázu vo formáte `udb`, ktorá je potrebná pre vyhľadávanie v druhom riadku. Druhý riadok je samotné porovnávanie dvojíc sekvencií. Prepínač `-db` špecifikuje databázu, voči ktorej bude prebiehať vyhľadávanie. Prepínač `-id` určuje, akú najmenšiu mieru identity musí dvojica vykazovať, aby bola zarátaná do výstupu. Zvolili sme hodnotu `0`, pretože sme skúmali celkovú podobnosť sekvencií v rámci sady a zaujímali nás teda ako vysoké, tak aj nízke miery identity. Prepínač `-self` oznamuje programu `usearch`, aby ignoroval dvojice s rovnakou hlavičkou (teda aby nezobrazoval identitu sekvencie so sebou samou ale zobrazil prípady, keď dva rôzne záznamy budú mať rovnakú sekvenciu). Prepínač `-userout` špecifikuje, že požadujeme vlastný formát výstupu a taktiež definuje výstupný súbor. Prepínač `-userfields` následne definuje požadovaný výstupný formát, v našom prípade sme si nechali vypísať položky `query` (hlavička jednej sekvencie z dvojice), `target` (hlavička druhej sekvencie z dvojice) a `id` (miera identity). Posledný prepínač `-maxaccepts` nastavuje maximálny počet prijatých dvojíc, ktorý sme nastavili na 10. Ak teda jedna sekvencia bola podobná viac než desiatim iným sekvenciám, program vráti 10 dvojíc s najvyššou mierou identity. Implicitná hodnota tohto prepínača je 1, čo by našej analýze nevyhovovalo. Výsledky týchto porovnaní pre každú dátovú sadu osobitne sme následne vykreslili s využitím balíčkov `numpy` a `matplotlib`.

Po analýzach sme našu dátovú sadu pozhlukovali postupne až na úroveň 30% identity, s využitím nástroja *CD-hit* a *Metacentra*. Túto finálnu podobu dátovej sady sme ešte analyzovali na podobnosť sekvencií, rovnako ako pôvodné dátové sady. Následne sme vytvorili našu databázu podľa schémy na obrázku 4.4. Na to sme využili program `mysql`, kde sme prostredníctvom konzolového rozhrania vytvorili databázu, používateľa a pridelili tomuto používateľovi všetky práva nad danou databázou. Zvyšok práce s databázou prebiehal cez skripty jazyka Python s využitím balíčka `MySQLdb`. Týmito skriptami sme vytvorili jednotlivé tabuľky databázy a naplnili tabuľky, ktoré nevyžadovali žiadne predikcie. Na prípravu a analýzu dát sme vytvorili približne 45 skriptov, pozostávajúcich spolu z približne 3 812 riadkov kódu.

6.2 Získanie predikcií a testovanie prediktorov

Po vytvorení finálnej dátovej sady sme potrebovali získať zo štyroch vybraných prediktorov ich predikcie. Prediktor *ESPRESSO* bol on-line, ale autori poskytli skript v jazyku *Perl*, ktorý spracovával vstupné súbory, komunikoval so serverom a prijímal naspäť predikcie. Vstupné súbory však nemohli byť vo FASTA formáte, preto sme ich pred spustením predikcie museli upraviť na nasledujúci formát:

```
<ID>        AA        <seq>        SOL        Ecoli
```

kde separátorom jednotlivých položiek bol znak `\t`, teda tabulátor. Položka `<ID>` bola naše unikátne ID a položka `<seq>` bola aminokyselinová sekvencia. Ostatné položky, teda `AA`, `SOL` a `Ecoli` sú priamo reťazce, ktoré v súbore musia byť a naznačujú prediktoru, že poskytujeme

aminokyselinovú sekvenciu (`AA`), žiadame predikciu rozpustnosti (`SOL`, pretože prediktor vie predikovať aj expresiu) a požadujeme E. coli ako expresný systém.

Prediktor *SOLpro* je off-line a na vstupe očakáva súbor vo FASTA formáte. Zistili sme však, že pokiaľ daný vstupný súbor obsahuje viac záznamov, prediktor spracuje len prvý z nich. Navyše sme zistili, že predikcia na lokálnom stroji prebieha veľmi pomaly. Preto sme využili služby organizácie *Metacentrum* a predikcie nástroja *SOLpro* sme získavali na počítačoch tejto organizácie. Problém so spracovaním jediného záznamu sme vyriešili tak, že sme pomocou skriptu rozdelili vstupný súbor na stovky menších súborov, kde každý obsahoval jediný záznam, a po získaní každej predikcie sme výstup spracovali a pridali do vznikajúceho výstupného súboru. Tento po skončení všetkých behov programu *SOLpro* obsahoval všetky výsledky predikcie.

Prediktor, označený názvom *Davis et al.*, sme si museli vytvoriť sami na základe rovníc, ktoré zmieňovaní autori poskytli v článku. Tento prediktor sme napísali v jazyku C++ a ide o program, ktorý prijíma vstupný súbor vo FASTA formáte a spracuje ho. Spočíta výskyty určitých aminokyselín v sekvenciách a pre každú sekvenciu spočíta poskytnuté rovnice, ktorých výsledkom je jednak predikcia a jednak pravdepodobnosť tejto predikcie. Program každú vypočítanú predikciu zapíše do výstupného súboru. Zdrojový kód tohto programu obsahuje 80 riadkov čistého kódu.

Za najkomplikovanejší prediktor z pohľadu našich potrieb považujeme prediktor *PROSO II*. Problém spočíva v tom, že tento prediktor je dostupný len on-line, a na rozdiel napríklad od prediktoru *ESPRESSO* neposkytuje žiadne iné rozhranie, než grafické v internetovom prehliadači. Komunikovali sme teda s autorom prediktoru a na jeho podnet sme využili nástroj na testovanie internetových stránok, konkrétne *Selenium*. Tento nástroj sme využili v podobe balíčka do jazyka Python. Okrem tohto balíčka sme v danom skripte využili aj balíček `lxml` pre spracovanie XML a HTML.

Skript pracuje tak, že spustí internetový prehliadač, v našom prípade išlo o prehliadač *Mozilla Firefox* a načíta v ňom stránku prediktoru. Na stránke ďalej vyhľadá pole pre vkladanie sekvencií. Následne skript prechádza vstupný súbor vo FASTA formáte, z ktorého získa hlavičku a sekvenciu. Túto sekvenciu najprv skontroluje na neplatné znaky a následne ju vpíše do poľa na stránke. Skript vyhľadá tlačidlo na odoslanie a sekvenciu odošle. Potom čaká na zmenu na stránke, kde sa po skončení predikcie vygeneruje tabuľka s výsledkom. Text z tabuľky je získaný a spracovaný a sú z neho vytiahnuté informácie o výslednej predikcii a jej presnosti. Tabuľka je následne odstránená z DOM modelu, aby bolo v ďalšom behu možné detekovať zmenu na stránke (bez odstránenia tabuľky by totiž po odoslaní ďalšej sekvencie a obdržaní nového výsledku nenastala na stránke z pohľadu DOM modelu žiadna zmena a neboli by sme teda schopní detekovať ďalšiu predikciu). Výsledok predikcie je spolu s pôvodnou hlavičkou zapísaný do výstupného súboru. Po ukončení poslednej predikcie je prehliadač zatvorený a skript končí.

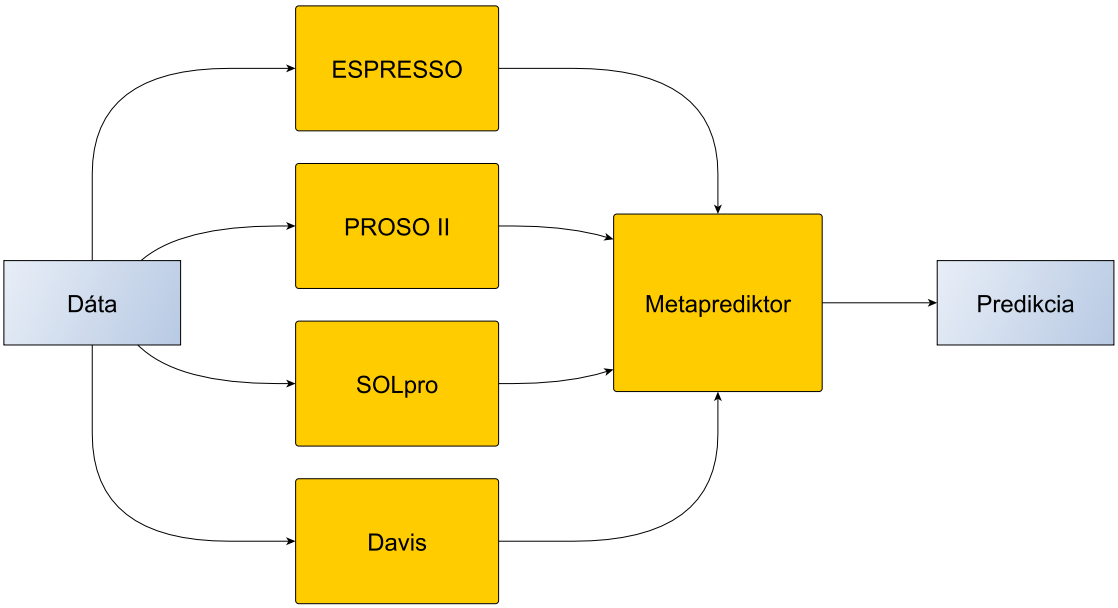
Výstupom každého vyššie spomenutého procesu bol výstupný súbor s hlavičkami a výsledkami predikcie pre každý prediktor. Tieto štyri výstupné súbory sme následne skriptom spracovali a informácie v nich obsiahnuté sme vložili do odpovedajúcich tabuliek našej databázy. Tým bola naša databáza úplná a obsahovala všetky potrebné informácie pre tvorbu metaprediktoru.

Pre účely testovania prediktorov bolo vhodné použiť vyvážené dátové sady, teda také, kde je počet rozpustných a nerozpustných proteínov rovnaký. Spočítali sme teda pre každý prediktor osobitne počet rozpustných proteínov, ku ktorým poznáme ich predikcie. Počet nerozpustných proteínov, ku ktorým nástroje poskytli predikcie, bol vždy väčší, než rozpustných. Pre každý prediktor osobitne sme potom vyradili náhodné nerozpustné proteíny

tak, aby počet rozpustných a nerozpustných proteínov bol rovnaký, a na tejto vyváženej sade sme spočítali presnosti jednotlivých prediktorov.

Následne sme zistili, pre ktoré sekvencie máme predikcie od všetkých štyroch prediktorov a vytvorili z týchto sekvencií novú dátovú sadu. Túto sme následne vyvážili odstránením náhodných nerozpustných sekvencií. Na tejto dátovej sade sme dodatočne otestovali všetky štyri prediktory. V súvislosti so získavaním predikcií a s testovaním prediktorov sme napísali približne 17 skriptov, obsahujúcich približne 601 riadkov kódu. Toto nezahŕňa C++ kód pre prediktor *Davis et al.*

6.3 Tvorba metaprediktorov



Obr. 6.2: Princíp metaprediktoru.

Obrázok 6.2 znázorňuje princíp nášho metaprediktoru. Dáta teda vstupujú paralelne do všetkých prediktorov prvého rádu a ich predikcie slúžia ako vstupy metaprediktoru, ktorý už produkuje finálnu predikciu. Je dobré podotknúť, že nie vždy musia byť dostupné predikcie od všetkých štyroch primárnych prediktorov. Je možné, či už kvôli výpadku online služby alebo iným problémom, že niektorý prediktor neposkytne predikciu pre danú sekvenciu. Metaprediktor sa s touto stratou dokáže vyrovnať. Pri tvorbe metaprediktoru sme využili tri prístupy.

V prvom rade sme implementovali v jazyku C++ doprednú neurónovú sieť, pracujúcu na princípe *back-propagation*. Bolo nutné zistiť optimálnu topológiu siete, teda počet vrstiev a počet neurónov v jednotlivých vrstvách. Použili sme teda našu finálnu dátovú sadu, resp. jej vyváženú podmnožinu, pre krížovú validáciu nášho prediktoru s rôznymi topológiami. Najvyššiu presnosť dosiahla sieť s jedinou skrytou vrstvou, kde vstupná vrstva prirodzene obsahovala toľko neurónov, koľko položiek mal vstupný vektor, skrytá vrstva mala 17 neurónov a potom následovala výstupná vrstva s jediným neurónom. Presnosť tejto siete bola 55,2%, je nutné však podotknúť, že iné topológie neboli príliš odlišné, presnosť sa vždy po-

hybovala okolo 54%. Kód pre neurónovú sieť dosiahol počet 455 riadkov kódu (+91 riadkov kódu v hlavičkovom súbore).

Uvedená presnosť je celkom nízka, z čoho sme usúdili, že možno obsahuje naša implementácia siete nejakú chybu. Využili sme teda framework *WEKA*[2] v nádeji, že modely, implementované v tomto frameworku, prinesú vyššiu presnosť. Pre trénovanie modelov bolo nutné trénovaciu sadu upraviť do formátu *ARFF*, naša trénovacia sada pre modely *WEKA* potom vyzerala nasledovne:

```
@RELATION Solubilities_features

@ATTRIBUTE solDavis NUMERIC
@ATTRIBUTE probDavis NUMERIC
@ATTRIBUTE solESPRESSOProp NUMERIC
@ATTRIBUTE probESPRESSO NUMERIC
@ATTRIBUTE solESPRESSOMotif NUMERIC
@ATTRIBUTE solPROSO NUMERIC
@ATTRIBUTE probPROSO NUMERIC
@ATTRIBUTE solSOLpro NUMERIC
@ATTRIBUTE probSOLpro NUMERIC
@ATTRIBUTE sol {soluble,insoluble}
@DATA
1,0.618309,0,0.534,1,?,?,?,?,soluble
0,0.825298,1,0.531,0,?,?,?,?,soluble
```

Prvý riadok špecifikuje názov vzťahov v danom súbore a ide prakticky len o informačný názov, ktorý nijak neovplyvňuje samotnú predikciu. Riadky, začínajúce kľúčovým slovom `@ATTRIBUTE` predstavujú vstupné či výstupné atribúty predikcie. Za týmto slovom nasleduje názov atribútu a za ním jeho typ, v našom prípade sú všetky vstupné atribúty numerické. Posledný riadok `@ATTRIBUTE` obsahuje výstupnú triedu predikcie a ide o nominálny atribút, nadobúdajúci dve možné hodnoty. Riadok `@DATA` oznamuje, že všetky zvyšné riadky obsahujú samotné trénovacie dáta, alebo dáta, na ktorých ma prebiehať predikcia. Každý takýto riadok obsahuje čiarkou oddelené hodnoty, odpovedajúce atribútom, uvedeným vyššie, v danom poradí. Nedostupnosť daného atribútu v danej dátovej položke je reprezentovaná otáznikom. Ako sme spomenuli, posledný atribút je výstupná trieda, v našom uvedenom príklade patria obidva uvedené vzorky medzi rozpustné proteíny. Keď je tento atribút dodaný, *WEKA* to chápe ako trénovaciu vzorku. Pokiaľ namiesto tohto atribútu dodáme otáznik, *WEKA* chápe danú vzorku ako kandidáta na predikciu a výslednú triedu teda po natrénovaní modelu predikuje.

Po vytvorení trénovacej množiny vo formáte *ARFF* sme vyskúšali rôzne modely, ktoré framework *WEKA* poskytuje. Použili sme 14 rôznych modelov, založených na rôznych metódach, od logistickej regresie cez rozhodovacie stromy a neurónové siete až po *SVM*. Najvyššiu presnosť dosiahla metóda *LMT* (*logistic model tree*). Princíp metódy *LMT* spočíva v kombinácii rozhodovacích stromov a logistickej regresie. Detailnejšie sa o metóde zmieňujeme v nasledujúcej časti. Na vyváženej dátovej sade sme teda natrénovali model *LMT*.

Tento prediktor má podobu skriptu v jazyku *Python* a funguje na nasledujúcom princípe: používateľom zadanú sekvenciu (či už zo súboru alebo priamo vloženú ako parameter príkazového riadku) prepošle paralelne (s využitím balíčka `multiprocessing`) do všetkých

[2] http://www.cs.waikato.ac.nz/ml/weka/

štyroch prediktorov a počká na ich výsledky. Tieto prediktory majú podobu samostatných skriptov a programov, ktoré sú volané z hlavného skriptu. Keďže vstupom do metaprediktoru môže byť súbor, obsahujúci viacero záznamov, predávame každému prediktoru názov vstupného súboru a každý prediktor si tento súbor spracováva osobitne. Tým sme dosiahli nezávislosť prediktorov, kde každý môže ísť svojou rýchlosťou a nemusí napríklad čakať na dokončenie ostatných prediktorov. Každý prediktor teda všetky svoje predikcie zapíše do svojho výstupného súboru. Tieto štyri súbory následne použijeme ako vstupy druhej fázy prediktoru, teda *LMT* z frameworku *WEKA*. Tu sa využíva už natrénovaný model, ktorý sme vytvorili z našej dátovej sady. Tento model vyžaduje vstupné dáta vo formáte *ARFF*[3], preto skript získané predikcie ešte prevedie do požadovanej podoby a až potom ich zašle do samotného modelu *LMT*. Výstup z nástroja *WEKA* následne náš skript spracuje a na štandardný výstup vypíše výsledok v nasledujúcom formáte:

```
<header>
<pred>        <prob>
```

kde `<header>` je pôvodná hlavička danej sekvencie zo vstupného súboru, `<pred>` je predikovaná rozpustnosť (nadobúdajúca hodnoty `soluble` a `insoluble`) a `<prob>` je pravdepodobnosť predikcie. Položky `<pred>` a `<prob>` sú oddelené tabulátorom. Uvedené dva riadky predstavujú jeden záznam, pokiaľ bolo na vstupe viac záznamov, po týchto dvoch riadkoch nasledujú ďalšie v rovnakom formáte. Pre spúšťanie samostatných skriptov a programov z hlavného skriptu sme využili balíček `subprocess`. Tento prediktor sme si pracovne označili ako `pred_weka`.

Posledným prístupom, ktorý sme vyskúšali, bola tvorba akéhosi majoritného metaprediktoru. Tento prediktor pracuje tak, že sa pozrie na predikcie vstupných prediktorov a ich pravdepodobnosti, ktoré použije ako akési váhy pri budovaní konsenzu. Konsenzus sa buduje tak, že na počiatku si prediktor nastaví výsledok na hodnotu 0. Pokiaľ niektorý vstupný prediktor predikuje sekvenciu ako rozpustnú, pripočíta metaprediktor jeho pravdepodobnosť k vznikajúcemu výsledku. Pokiaľ predikuje vstupný prediktor sekvenciu ako nerozpustnú, je jeho pravdepodobnosť od vznikajúceho výsledku odčítaná. Po započítaní všetkých predikcií je teda vytvorené jediné číslo v rozsahu od -4 do 4 (pretože pokiaľ každý vstupný prediktor predikoval sekvenciu ako nerozpustnú s maxinálnou pravdepodobnosťou, teda 1, výsledok bude $0 - 1 - 1 - 1 - 1$, analogicky keby všetky prediktory predikovali sekvenciu ako rozpustnú). Na základe tohto čísla je vytvorená finálna predikcia.

Väčšina tohto prediktoru je zhodná s predchádzajúcim prediktorom `pred_weka`. Aj tento prediktor preposiela vstupný súbor ostatným prediktorom, ktoré volá ako samostatné skripty a programy a následne spracováva ich výstupy. Rozdiel je v tom, že tento prediktor nevyužíva framework *WEKA*, ale spracováva výsledky primárnych predikcií spôsobom, uvedeným vyššie. Výstup tohto prediktoru je podobný ako výstup predchádzajúceho, až na to, že tento prediktor neposkytuje na výstupe pravdepodobnosť predikcie, výstup teda obsahuje len položky `<header>` a `<pred>`. Tento prediktor sme si pracovne označili ako `pred_major`. Pri tvorbe metaprediktorov a súvisiacich úkonoch vzniklo spolu približne 10 skriptov s obsahom približne 689 riadkov kódu. V týchto číslach nie je zahrnutý vyššie spomenutý C++ program, implementujúci neurónovú sieť.

[3] http://www.cs.waikato.ac.nz/ml/weka/arff.html

6.3.1 LMT

Metóda *logistic model trees* bola predstavená v článku [21]. Populárnymi metódami pre klasifikáciu sú lineárna logistická regresia a rozhodovacie stromy, pričom každá z týchto metód poskytuje isté výhody a nevýhody. Lineárna logistická regresia využíva lineárny model na rozdelenie dát a vykazuje nízke riziko preučenia, na druhú stranu však hrozí, že model nespozná relevantné vzťahy medzi rysmi a dochádza k podučeniu (angl. *underfitting*). Rozhodovacie stromy naopak prehľadávajú široký priestor a dokážu zachytiť nelineárne vzory v dátach, vďaka čomu však vzniká riziko preučenia. Vhodnosť využitia jednej alebo druhej metódy teda závisí na veľkosti a druhu dát [21].

LMT pozostáva z klasického rozhodovacieho stromu, kde však listy obsahujú regresné funkcie. Rovnako ako v bežných rozhodovacích stromoch obsahujú všetky vnútorné uzly test na niektorý rys vstupných dát. V bežnom rohodovacom strome je listom uzol, ktorý obsahuje podmnožinu dát, ktorá obsahuje prevažnú väčšinu dát rovnakej triedy. Pokiaľ sa pri klasifikácii nových dát dôjde do listového uzlu, sú tieto dáta klasifikované do tej triedy, ktorá je väčšinová v danom uzle. V pripade *LMT* sú na mieste listov regresné funkcie, pričom každá takáto funkcia používa ako svoje vstupy rysy z podmnožiny pôvodných všetkých rysov pre dané dáta. Táto funkcia na výstupe generuje pravdepodobnosť príslušnosti dát k určitej triede [21].

Tvorba tohto typu stromu prebieha podobne, ako tvorba klasického rozhodovacieho stromu s tým rozdielom, že v každom uzle je vybudovaný model logistickej regresie na podmnožine dát, ktorá danému uzlu odpovedá. Budovanie modelu v každom uzle je nutné, pretože strom bude neskôr podliehať prerezávaniu (angl. *pruning*) (ako sme spomenuli v časti 3.3.2) a každý uzol je potenciálnym listom finálneho stromu. Namiesto tvorby nových regresných modelov v každom uzle sú pre synovské uzly využité regresné modely rodičovských uzlov, ktoré sú dodatočne zjemnené použitím rysov, ktoré sú platné v konkrétnej podmnožine pôvodných dát, ktorá zodpovedá danému uzlu [21].

Princíp modelu teda spočíva v tom, že pokiaľ je na vstupe veľké množstvo dát, rozhodovací strom tieto dáta rozdeľuje na menšie časti. V určitom momente už sú podmnožiny dát dostatočne malé na to, aby logistická regresia dokázala zachytiť podstatné rysy pre klasifikáciu dát, a delenie stromu končí listom s regresnou funkciou.

6.4 Testovanie metaprediktorov

Pre trénovanie prediktoru `pred_weka` sme využili našu dátovú sadu, resp. jej vyváženú podmnožinu. Prediktor bol teda trénovaný (a testovaný) na sade so 45 114 sekvenciami, z ktorých práve polovica bola rozpustná. Testovanie prebiehalo pomocou 10-násobnej krížovej validácie, ktorú vykonal samotný framework *WEKA* pri trénovaní modelu. Toto testovanie ukázalo presnosť predikcie 58,83% a plochu pod ROC krivkou 0,621.

Dodatočne sme chceli prediktor otestovať aj na dátach, ktoré neboli pri trénovaní použité. Na stránke prediktoru *ccSOL omics* sme získali sekvencie, rozdelené do štyroch skupín: *chap_dep*, *aggreg*, *fold* a *chap_indep*. Tieto sekvencie bohužiaľ neboli anotované ich reálnou rozpustnosťou, je možné však predpokladať, že napríklad skupina sekvencií zo sady *chap_dep* bude prevažne nerozpustná vzhľadom na ich závislosť na chaperonoch (*chaperone dependent*), podobne aj sada *aggreg* by mala prevažne tvoriť inklúzie a teda by väčšina mala byť nerozpustná, naopak sady *fold* a *chap_indep* by mali obsahovať prevažne rozpustné sekvencie. Z týchto štyroch sád sme prirodzene vylúčili sekvencie, ktoré sa nachádzali aj v trénovacej sade.

Vzhľadom na absenciu presných anotácií nemôžeme na týchto sadách merať presnosť, avšak dobre pracujúci prediktor by mal predikovať dané sekvencie tak, ako sme uviedli pred chvíľou, teda napríklad by mal zaradiť väčšinu sekvencií zo sady *chap_dep* medzi nerozpustné. S touto myšlienkou sme otestovali náš prediktor `pred_weka` na týchto štyroch sadách a výsledky ukazujeme v tabuľke 6.1. Ukazuje sa, že napriek trénovaniu na vyváženej dátovej sade je prediktor značne naklonený nerozpustným proteínom a predikuje do tejto triedy takmer všetky sekvencie.

Dátová sada	Predikované ako rozpustné	Predikované ako nerozpustné
chap_dep	8	171
aggreg	8	41
fold	11	58
chap_indep	5	29

Tabuľka 6.1: Testovanie metaprediktoru `pred_weka`.

Pri testovaní prediktoru `pred_major` sme mali širšie možnosti, keďže tento prediktor v princípe nevyužíva strojové učenie a nebolo teda nutné ho na žiadnych dátach trénovať, vďaka čomu sme mohli využiť celú našu dátovú sadu na testovanie. Z našej sady sme opäť vybrali vyváženú podmnožinu a testovali sme teda prediktor na 45 114 sekvenciách. Testy ukázali presnosť predikcie 52,41% a MCC 0,052. Dodatočne sme prediktor otestovali aj na štyroch neanotovaných sadách, rovnako ako prediktor `pred_weka` a výsledky ukazujeme v tabuľke 6.2. Tento prediktor prekvapivo poskytol pri tomto testovaní lepšie výsledky, ktoré presnejšie zodpovedajú očakávaným výsledkom. Predikcie tu nie sú zavážené smerom k nerozpustnej triede a sady *chap_dep* a *fold* sú medzi triedy rozdelené podľa nášho názoru veľmi rozumne. Pri sade *chap_indep* je však predikcia značne neistá.

Dátová sada	Predikované ako rozpustné	Predikované ako nerozpustné
chap_dep	49	130
aggreg	22	27
fold	64	5
chap_indep	15	19

Tabuľka 6.2: Testovanie metaprediktoru `pred_major`.

Na záver poskytujeme prehľadnú tabuľku presností našich prediktorov (6.3). Pre testovanie našich metaprediktorov a súvisiace úlohy sme vytvorili približne 5 skriptov, ktoré tvorí spolu približne 256 riadkov kódu.

Prediktor	Presnosť	MCC	Plocha pod ROC krivkou
`meta_weka`	58,83%	0,179	0,621
`meta_major`	52,41%	0,052	-

Tabuľka 6.3: Presnosti metaprediktorov.

Kapitola 7

Záver

Preskúmali sme princípy tvorby rekombinantných proteínov a existujúce algoritmy pre predikciu rozpustnosti týchto proteínov. Predstavili sme taktiež rôzne metódy strojového učenia, používané okrem iného v oblasti predikcie rozpustnosti proteínov. Pripravili sme dátovú sadu z rôznych dátových sád, použitých tvorcami už existujúcich prediktorov, pričom sme pôvodné dátové sady podrobili analýze. Z mnohých prediktorov prezentovaných v článku [12] sme opísali tie, ktoré považujeme za významné, a následne využili tie, ktoré sú dostupné a použiteľné pre našu prácu. Tieto prediktory sme otestovali na našich dátach a poskytli sme aj rozbor dôvodu ich presností.

V poslednej časti sme vybudovali dva konsenzuálne prediktory, ktoré využili predikcie primárnych prediktorov. Vyskúšali sme viaceré metódy strojového učenia a výsledné metaprediktory otestovali na rôznych dátach. Konsenzuálny prediktor z princípu závisí na výstupoch a presnostiach vstupných prediktorov. Berie do úvahy rôzne predikcie a vďaka využitiu viacerých rôznych pohľadov na daný problém zvyšuje presnosť predikcie. Keď sa pozrieme na priemernú presnosť našich vstupných prediktorov, môžeme tvrdiť, že naše metaprediktory poskytujú vo výsledku vyššiu presnosť. Z celkového pohľadu je však dosiahnutá presnosť veľmi nízka a blíži sa náhodnému výberu výslednej triedy.

Teoreticky sa teda podarilo zvýšiť presnosť predikcie, z praktického hľadiska je však presnosť príliš nízka na reálne použitie, a to práve z dôvodu nízkych presností vstupných prediktorov. Konsenzuálna predikcia teoreticky poskytuje vyššiu presnosť, avšak závisí aj na kvalitných prediktoroch prvej úrovne. Zdá sa teda, že oblasť predikcie rozpustnosti proteínov nie je ešte dostatočne rozvinutá a presná na to, aby v nej našla konsenzuálna predikcia uplatnenie.

Do budúcna sa teda zdá podstatnejší vývoj samotných prediktorov prvej úrovne, z ktorých by potencionálne budúce metaprediktory mohli ťažiť. V tejto práci boli spomenuté rôzne prístupy k predikcii a boli ukázané rysy aminokyselinových sekvencií, ktoré dobre odrážajú rozpustnosť proteínu a pravdepodobnosť jeho úspešného zloženia. Na základe týchto informácií by bolo možné s využitím techník strojového učenia vybudovať nový prediktor rozpustnosti.

Ako problém sa v tejto oblasti ukázal aj zber vhodných dát, na čo poukázalo viacero autorov vo svojich vlastných prácach. Dobrou cestou ku skvalitneniu predikcie by teda rozhodne bol pečlivý zber a anotácia sekvencií v rámci jednotného, dobre zdokumentovaného protokolu. Kvalitné a dobre anotované dáta z jednotných zdrojov by poskytli vhodné zázemie pre tvorbu presných prediktorov rozpustnosti.

Literatúra

[1] Agostini, F.; Cirillo, D.; Livi, C. M.; aj.: ccSOL omics: a webserver for solubility prediction of endogenous and heterologous expression in Escherichia coli. *Bioinformatics*, ročník 30, č. 20, 2014.

[2] Agostini, F.; Vendruscolo, M.; Tartaglia, G. G.: Sequence-Based Prediction of Protein Solubility. *Journal of Molecular Biology*, ročník 421, č. 2-3, 2012.

[3] Alpaydin, E.: *Introduction to Machine Learning (2nd edition)*. The MIT Press, 2009, ISBN 978-0262012430.

[4] Baldi, P.; Brunak, S.: *Bioinformatics: The Machine Learning Approach, Second Edition*. A Bradford Book, 2001, ISBN 0-262-02506-X.

[5] Carter, M.; Shieh, J. C.: *Guide to Research Techniques in Neuroscience*. Academic Press, 2015, ISBN 9780128005972.

[6] Chan, W.-C.; Liang, P.-H.; Shih, Y.-P.; aj.: Learning to predict expression efficacy of vectors in recombinant protein production. *BMC Bioinformatics*, ročník 11, č. 2, 2010.

[7] Davis, G. D.; Elisee, C.; Newham, D. M.; aj.: New Fusion Protein Systems Designed to Give Soluble Expression in Escherichia coli. *Biotechnology and Bioengineering*, ročník 65, č. 4, 1999.

[8] Diaz, A. A.; Tomba, E.; Lennarson, R.; aj.: Prediction of Protein Solubility in Escherichia coli Using Logistic Regression. *Biotechnology and Bioengineering*, ročník 105, č. 2, 2010.

[9] Fang, Y.; Fang, J.: Discrimination of soluble and aggregation-prone proteins based on sequence information. *Molecular BioSystems*, ročník 9, č. 4, 2013.

[10] Fu, L.; Niu, B.; Zhu, Z.; aj.: CD-HIT: accelerated for clustering the next generation sequencing data. *Bioinformatics*, ročník 28, č. 23, 2012.

[11] Goh, C.-S.; Lan, N.; Douglas, S. M.; aj.: Mining the StrucStructural Genomics Pipeline: Identification of Protein Properties that Affect High-throughput Experimental Analysis. *Journal of Molecular Biology*, ročník 336, č. 1, 2004.

[12] Habibi, N.; Hashim, S. Z. M.; Norouzi, A.; aj.: A review of machine learning methods to predict the solubility of overexpressed recombinant proteins in Escherichia coli. *BMC Bioinformatics*, ročník 15, č. 134, 2014.

[13] Herculano-Houzel, S.: The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in Human Neuroscience*, ročník 3, č. 31, 2009.

[14] Hirose, S.; Kawamura, Y.; Yokota, K.; aj.: Statistical analysis of features associated with protein expression/solubility in an in vivo Escherichia coli expression system and a wheat germ cell-free expression system. *Journal of biochemistry*, ročník 150, č. 1, 2011.

[15] Hirose, S.; Noguchi, T.: ESPRESSO: A system for estimating protein expression and solubility in protein expression systems. *PROTEOMICS*, ročník 13, č. 9, 2013.

[16] Ho, T. K.: Random Decision Forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, ročník 1, 1995, s. 278–282.

[17] Huang, H.-L.; Charoenkwan, P.; Kao, T.-F.; aj.: Prediction and analysis of protein solubility using a novel scoring card method with dipeptide composition. *BMC Bioinformatics*, ročník 13, č. 17, 2012.

[18] Idicula-Thomas, S.; Balaji, P. V.: Understanding the relationship between the primary structure of proteins and its propensity to be soluble on overexpression in Escherichia coli. *Protein Science*, ročník 14, č. 3, 2005.

[19] Idicula-Thomas, S.; Kulkarni, A. J.; Kularni, B. D.; aj.: A support vector machine-based method for predicting the propensity of a protein to be soluble or to form inclusion body on overexpression in Escherichia coli. *Bioinformatics*, ročník 22, č. 3, 2006.

[20] Kocbek, S.; Stiglic, G.; Pernek, I.; aj.: Stability of different feature selection methods for selecting protein sequence descriptors in protein solubility classification problem. In *2010 IEEE 2šrd International Symposium on Computer-Based Medical Systems (CBMS)*, ročník 1, 2010, s. 50–55.

[21] Landwehr, N.; Hall, M.; Frank, E.: Logistic Model Trees. *Machine Learning*, ročník 59, č. 1, 2005.

[22] Li, W.; Godzik, A.: Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, ročník 22, č. 13, 2006.

[23] Lukáš, R., Burgetová, I.: Klasifikace a predikce. Prednáška predmetu ZZN. URL https://www.fit.vutbr.cz/study/courses/ZZN/private/prednasky/06_klasifikace_predikce.pdf

[24] Magnan, C. N.; Randall, A.; Baldi, P.: SOLpro: accurate sequence-based prediction of protein solubility. *Bioinformatics*, ročník 25, č. 17, 2009.

[25] Nelson, D. L.; Lehninger, A. L.; Cox, M. M.: *Lehringer Principles of Biochemistry, Fourth Edition*. W. H. Freeman, 2004, ISBN 978-0716743392.

[26] Niwa, T.; Ying, B.-W.; Saito, K.; aj.: Bimodal protein solubility distribution revealed by an aggregation analysis of the entire ensemble of Escherichia coli proteins. *Proceedings of the National Academy of Sciences of the United States of America*, ročník 106, č. 11, 2009.

[27] Palomares, L. A.; Estrada-Mondaca, S.; Ramírez, O. T.: Production of recombinant proteins. In *Recombinant Gene Expression*, kapitola 2, Humana Press, 2004, ISBN 978-1-58829-262-9.

[28] Russell, S.; Norvig, P.: *Artificial Intelligence: A Modern Approach (2rd Edition)*. Prentice Hall, 2002, ISBN 978-0137903955.

[29] Russell, S.; Norvig, P.: *Artificial Intelligence: A Modern Approach (3rd Edition)*. Pearson, 2009, ISBN 978-0136042594.

[30] Samak, T.; Gunter, D.; Wang, Z.: Prediction of Protein Solubility in E. coli. In *2012 IEEE 8th International Conference on E-Science*, ročník 1, 2012, s. 1–8.

[31] Smialowski, P.; Doose, G.; Torkler, P.; aj.: PROSO II – a new method for protein solubility prediction. *The FEBS Journal*, ročník 279, č. 12, 2012.

[32] Smialowski, P.; Martin-Galiano, A. J.; Mikolajka, A.; aj.: Protein solubility: sequence based prediction and experimental verification. *Bioinformatics*, ročník 23, č. 19, 2007.

[33] Stiglic, G.; Kocbek, S.; Pernek, I.; aj.: Comprehensive Decision Tree Models in Bioinformatics. *PLoS ONE*, ročník 7, č. 3, 2012.

[34] Voet, D.: *Fundamentals of Biochemistry: Life at the Molecular Level (3rd Edition)*. Wiley, 2008, ISBN 978-0470129302.

[35] Wilkinson, D. L.; Harrison, R. G.: Predicting the solubility of recombinant proteins in Escherichia coli. *Nature Biotechnology*, ročník 9, č. 5, 1991.

[36] WWW stránky: ECOCYC accession numbers. [cit. 2015-12-25].
URL http://ecocyc.org/EcoCycUserGuide.shtml

[37] Xiaohui, N.; Feng, S.; Xuehai, H.; aj.: Predicting the protein solubility by integrating chaos games representation and entropy in information theory. *Expert Systems with Applications*, ročník 41, č. 4, 2014.